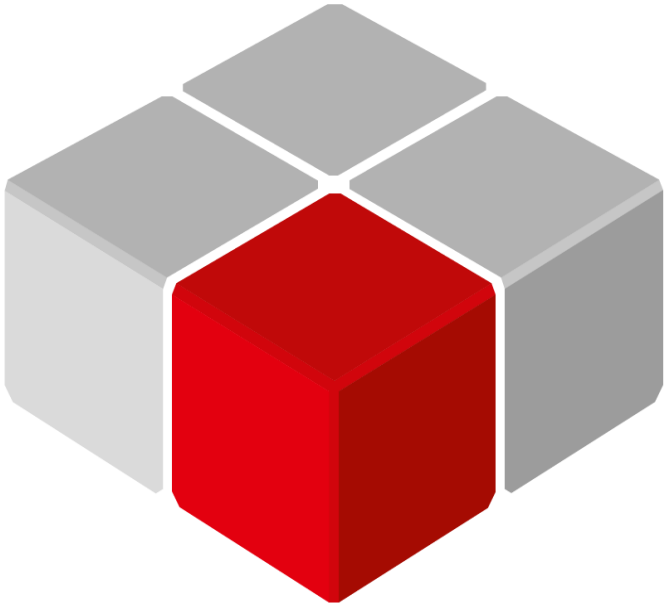
CODESYS V3.5

Описание библиотеки OwenStringUtils



Руководство пользователя

10.05.2022
версия 3.0

Оглавление

1 Цель документа

Настоящее руководство представляет собой описание библиотеки **OwenStringUtils**, которая предоставляет пользователю дополнительный функционал для работы со строками – в частности, функции конвертации строк **ASCII** в строки **Unicode** и **Unicode** в **ASCII**. В данном документе описана версия библиотеки **3.5.4.9**.

ПРИМЕЧАНИЕ

Функции библиотеки позволяют работать со строками, длина которых не превышает **255** символов.

ПРИМЕЧАНИЕ

Базовые функции работы со строками содержатся в библиотеках **Standard**, **Standard64** и **StringUtils**, которые входят в состав **CODESYS**.

2 Описание библиотеки OwenStringUtils

2.1 Установка библиотеки

Для установки библиотеки в **CODESYS** в меню **Инструменты** следует выбрать пункт **Репозиторий библиотек**, нажать кнопку **Установить,** указать путь к библиотеке и нажать **Открыть**:

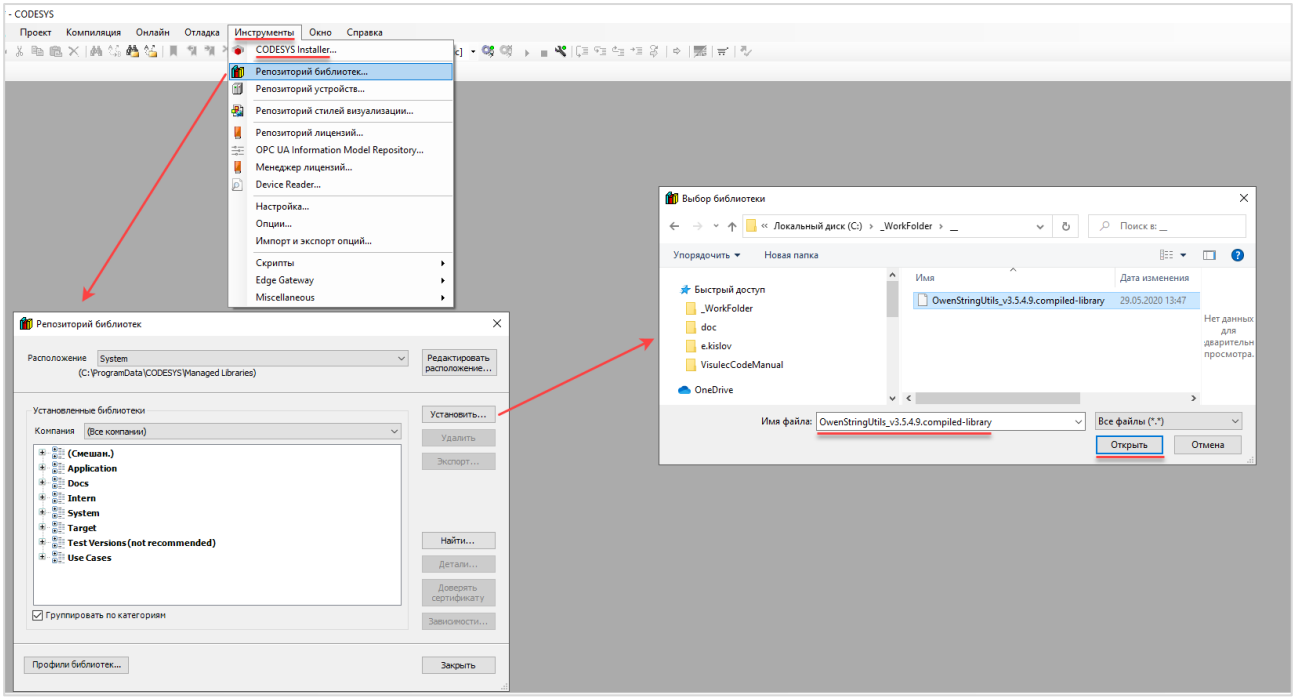


Рисунок 2.1 – Установка библиотеки в среду CODESYS

2.2 Добавление библиотеки в проект CODESYS

Для добавления библиотеки **OwenStringUtils** в проект **CODESYS** следует в **Менеджере библиотек** нажать кнопку **Добавить библиотеку** и в списке библиотек найти библиотеку **OwenStringUtils**, после чего нажать **ОК**.

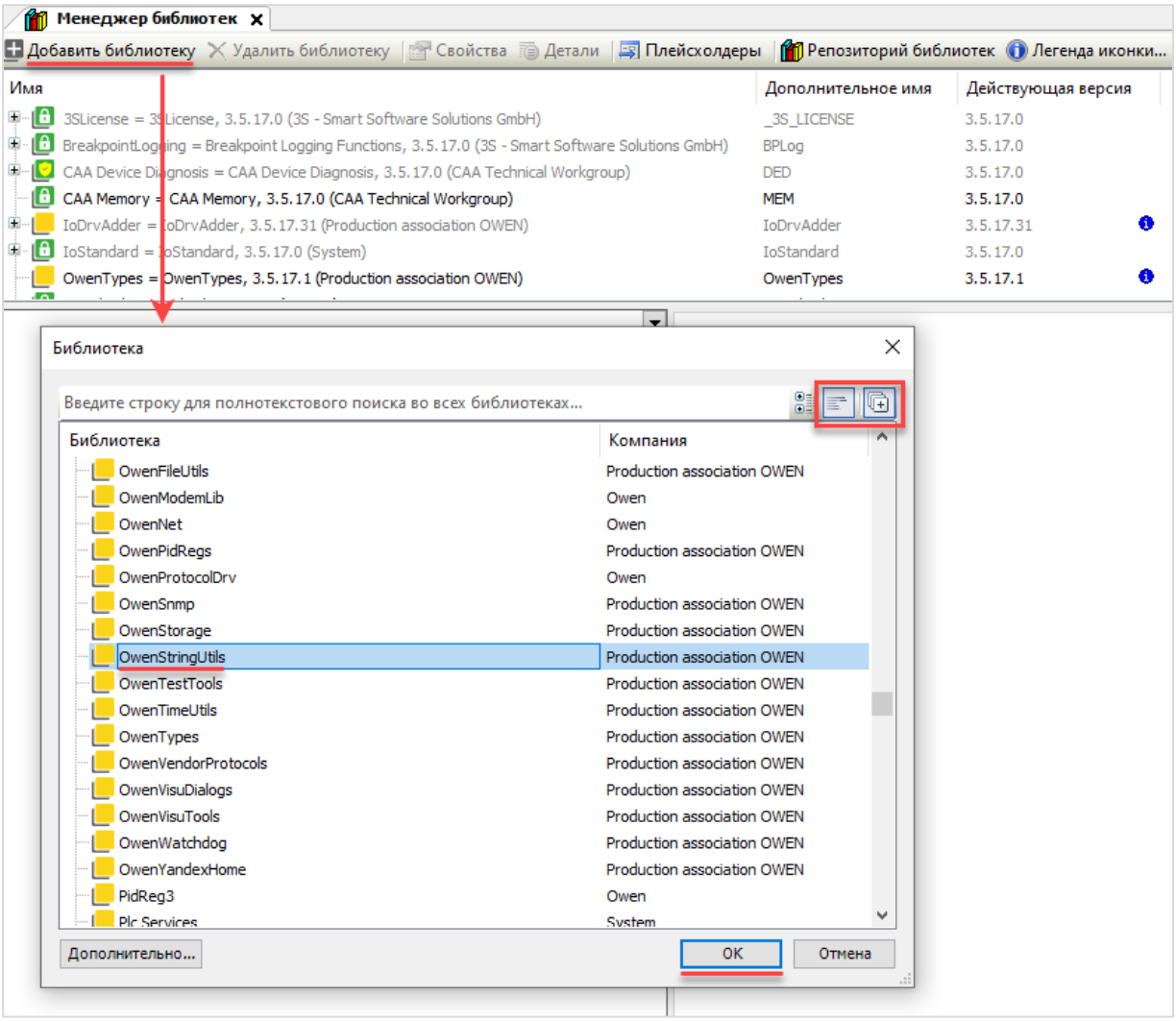


Рисунок 2.2 – Добавление библиотеки OwenStringUtils

После добавления библиотека появится в списке **Менеджера библиотек**:

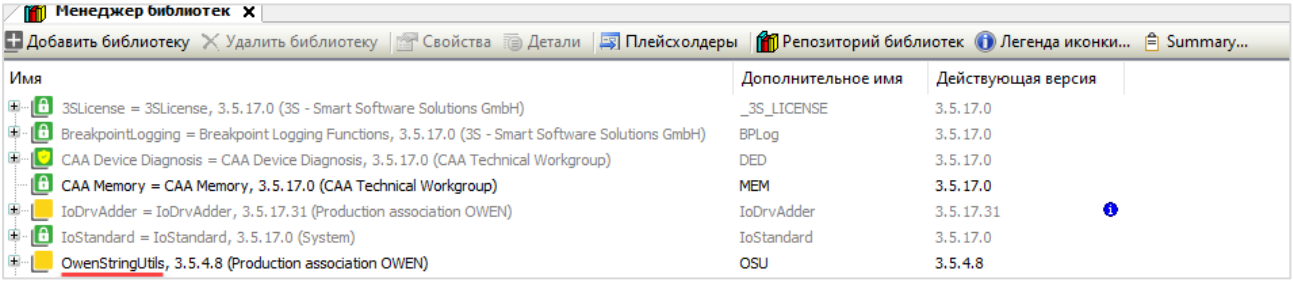


Рисунок 2.3 – Список библиотек проекта

ПРИМЕЧАНИЕ

При обращении к функциям библиотеки следует перед их названием указывать префикс **OSU** (пример: **OSU.After**).

3 Описание библиотеки

3.1 Преобразование кодировок

3.1.1 Функция CP1251_TO_UNICODE

Функция **CP1251_TO_UNICODE** используется для конвертации переменной типа **STRING**, содержащей строку в кодировке ASCII (CP1251), в переменную типа **WSTRING**, содержащую строку в кодировке Unicode (UCS-2).

Таблица 3.1.1 – Описание входов и выходов функции CP1251_TO_UNICODE

Имя переменной	Тип	Описание
Входные переменные		
sInputString	STRING(255)	Исходная строка в кодировке ASCII (CP1251)
Выходные переменные		
CP1251_TO_UNICODE	WSTRING(255)	Строка в кодировке Unicode (UCS-2)

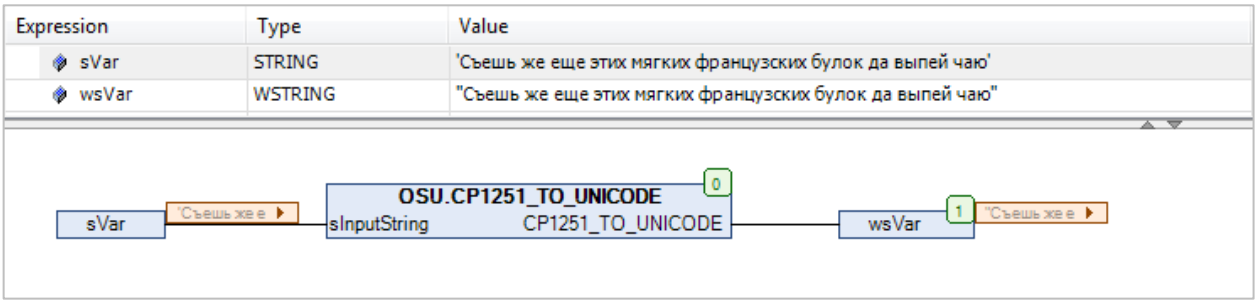


Рисунок 3.1.1 – Пример использования функции CP1251_TO_UNICODE на языке CFC

3.1.2 Функция UNICODE_TO_CP1251

Функция **UNICODE_TO_CP1251** используется для конвертации переменной типа **WSTRING**, содержащей строку в кодировке Unicode (UCS-2), в переменную типа **STRING**, содержащую строку в кодировке ASCII (CP1251). Если в исходной строке есть символы, отсутствующие в кодировке CP1251 – то они будут проигнорированы.

Таблица 3.1.2 – Описание входов и выходов функции UNICODE_TO_CP1251

Имя переменной	Тип	Описание
Входные переменные		
wsInputWstring	WSTRING(255)	Исходная строка в кодировке Unicode (UCS-2)
Выходные переменные		
UNICODE_TO_CP1251	STRING(255)	Строка в кодировке ASCII (CP1251)

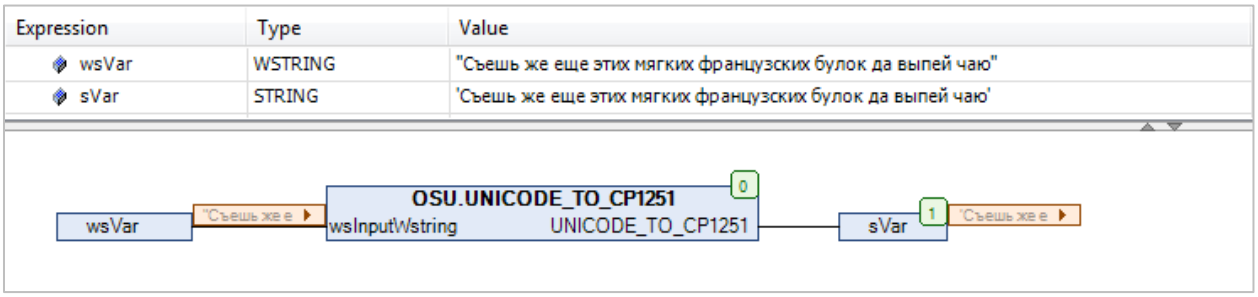


Рисунок 3.1.2 – Пример использования функции UNICODE_TO_CP1251 на языке CFC

3.2 Преобразование различных типов к STRING

3.2.1 Функция REAL_TO_STRING_FORMAT

Функция **REAL_TO_STRING_FORMAT** преобразует значение с плавающей точкой типа **REAL** в форматированную строку типа **STRING** с настраиваемым символом-разделителем целой/дробной части и количеством знаков после разделителя. Допустимые символы-разделители определяются перечислением **DECIMAL_SEPARATOR**. В случае выбора недопустимого символа в качестве разделителя используется точка. Если значение на входе превышает **1e8**, то функция возвращает строку с экспоненциальным представлением значения.

Таблица 3.2.1 – Описание входов и выходов функции REAL_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
rValueToConvert	REAL	Значение с плавающей точкой
usiSignificantDigitsCount	USINT	Количество знаков после разделителя
eDecimalSeparator	DECIMAL_SEPARATOR	Разделитель целой и дробной части
Выходные переменные		
REAL_TO_STRING_FORMAT	STRING(80)	Значение в виде форматированной строки

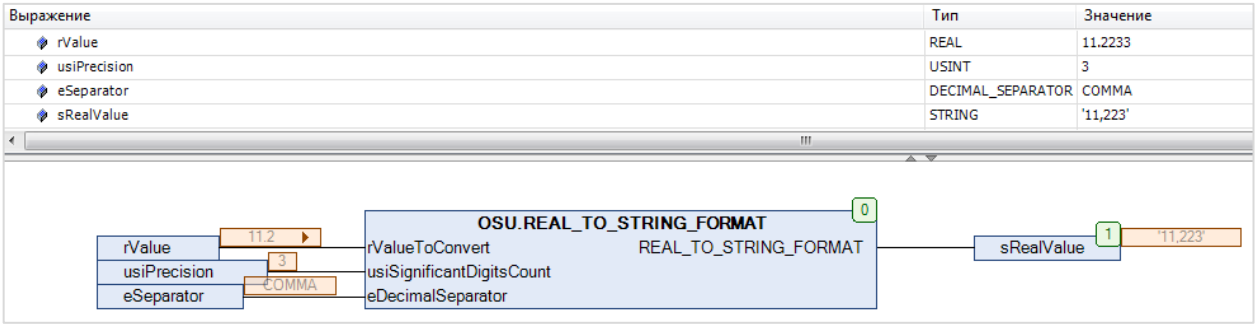


Рисунок 3.2.1 – Пример использования функции REAL_TO_STRING_FORMAT на языке CFC

Таблица 3.2.2 – Описание элементов перечисления DECIMAL_SEPARATOR

Название	Тип	Значение	Описание
DOT	USINT	0	Разделитель целой и дробной части – точка
COMMA	USINT	1	Разделитель целой и дробной части – запятая

3.2.2 Функция LREAL_TO_STRING_FORMAT

Функция **LREAL_TO_STRING_FORMAT** преобразует значение с плавающей точкой типа **LREAL** в форматированную строку типа **STRING** с настраиваемым символом-разделителем целой/дробной части и количеством знаков после разделителя. Допустимые символы-разделители определяются перечислением **DECIMAL_SEPARATOR**. В случае выбора недопустимого символа в качестве разделителя используется точка. Если значение на входе превышает **1e16**, то функция возвращает строку с экспоненциальным представлением значения.

Таблица 3.2.3 – Описание входов и выходов функции LREAL_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
lrValueToConvert	LREAL	Значение с плавающей точкой
usiSignificantDigitsCount	USINT	Количество знаков после разделителя
eDecimalSeparator	DECIMAL_SEPARATOR	Разделитель целой и дробной части
Выходные переменные		
LREAL_TO_STRING_FORMAT	STRING(80)	Значение в виде форматированной строки

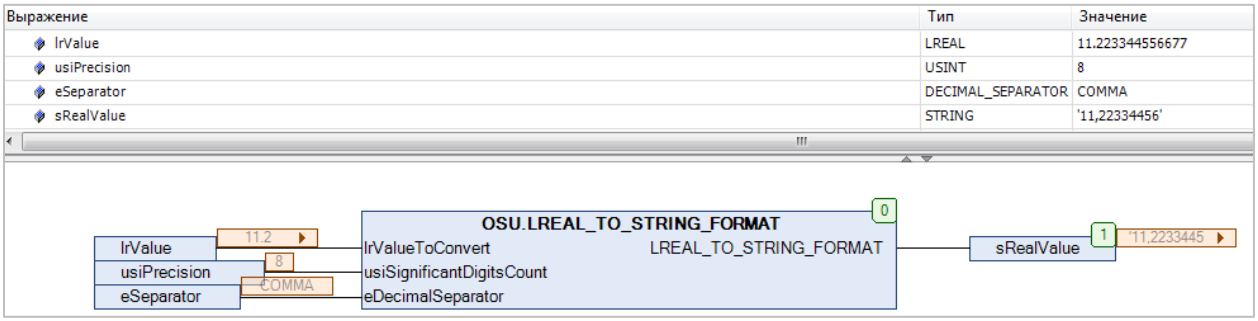


Рисунок 3.2.2 – Пример использования функции LREAL_TO_STRING_FORMAT на языке CFC

Таблица 3.2.4 – Описание элементов перечисления DECIMAL_SEPARATOR

Название	Тип	Значение	Описание
DOT	USINT	0	Разделитель целой и дробной части – точка
COMMA	USINT	1	Разделитель целой и дробной части – запятая

3.2.3 Функция DT_TO_STRING_FORMAT

Функция **DT_TO_STRING_FORMAT** заменяет в строке **sFormatString** первое вхождение подстроки типа %t[<заполнители>] на форматированное значение даты и времени **dtToConvert**. Список возможных заполнителей приведен в Приложении А. Все остальные символы строки **sFormatString** останутся без изменений. Если размер результирующей строки превышает 255 символов, то она будет обрезана до 255 символов.

Таблица 3.2.5 – Описание входов и выходов функции DT_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
dtToConvert	DT	Метка времени
sFormatString	STRING(255)	Форматированная строка с заполнителем метки времени
Выходные переменные		
DT_TO_STRING_FORMAT	STRING(255)	Форматированная строка с меткой времени

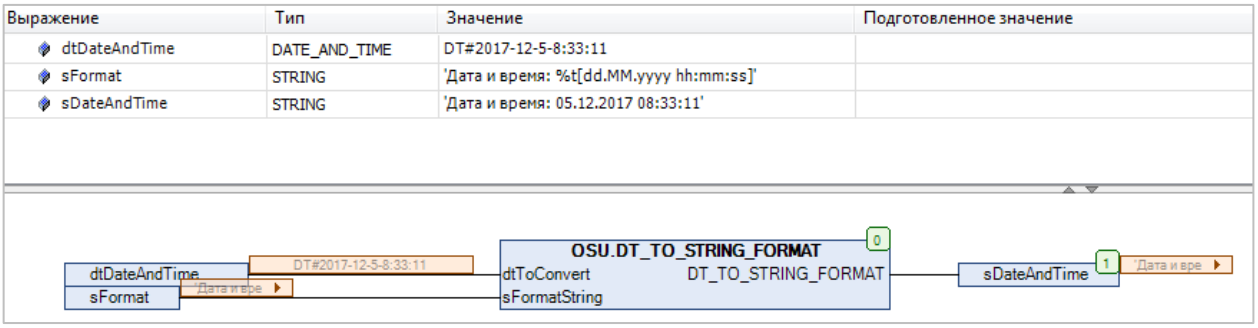


Рисунок 3.2.3 – Пример использования функции DT_TO_STRING_FORMAT на языке CFC

3.2.4 Функция DATE_TO_STRING_FORMAT

Функция **DATE_TO_STRING_FORMAT** заменяет в строке **sFormatString** первое вхождение подстроки типа %t[<заполнители>] на форматированное значение даты **dToConvert**. Список возможных заполнителей приведен в Приложении А. Все остальные символы строки **sFormatString** останутся без изменений. Если размер результирующей строки превышает 255 символов, то она будет обрезана до 255 символов.

Таблица 3.2.6 – Описание входов и выходов функции DATE_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
dDateToConvert	DATE	Дата
sFormatString	STRING(255)	Форматированная строка с заполнителем даты
Выходные переменные		
DATE_TO_STRING_FORMAT	STRING(255)	Форматированная строка с датой

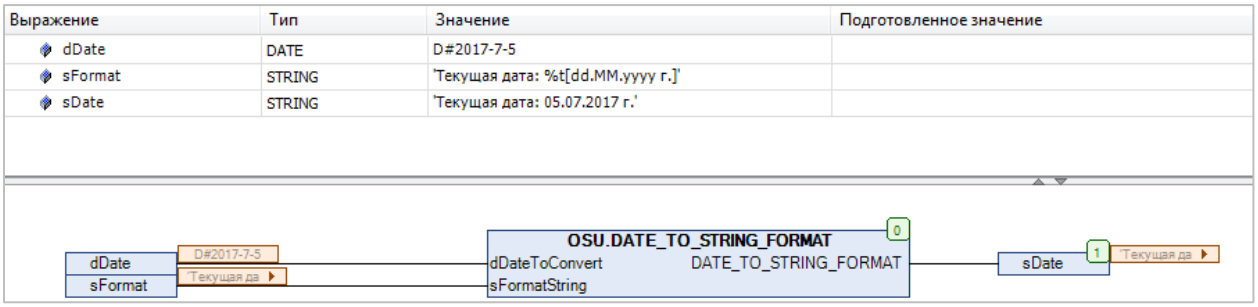


Рисунок 3.2.4 – Пример использования функции DATE_TO_STRING_FORMAT на языке CFC

3.2.5 Функция TOD_TO_STRING_FORMAT

Функция **TOD_TO_STRING_FORMAT** заменяет в строке **sFormatString** первое вхождение подстроки типа %t[<заполнители>] на форматированное значение времени суток **todToConvert**. Список возможных заполнителей приведен в Приложении А. Все остальные символы строки **sFormatString** останутся без изменений. Если размер результирующей строки превышает 255 символов, то она будет обрезана до 255 символов.

Таблица 3.2.7 – Описание входов и выходов функции TOD_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
todToConvert	TOD	Время суток
sFormatString	STRING(255)	Форматированная строка с заполнителем времени суток
Выходные переменные		
TOD_TO_STRING_FORMAT	STRING(255)	Форматированная строка с датой

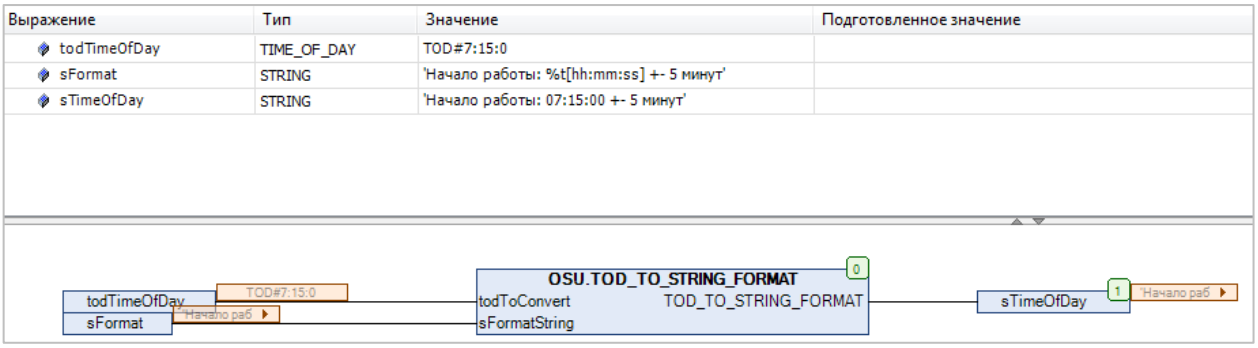


Рисунок 3.2.5 – Пример использования функции TOD_TO_STRING_FORMAT на языке CFC

3.2.6 Функция TIME_TO_STRING_FORMAT

Функция **TIME_TO_STRING_FORMAT** заменяет в строке **sFormatString** первое вхождение подстроки типа %t[<заполнители>] на форматированное значение времени **tTimeToConvert**. Список возможных заполнителей приведен в Приложении А. Все остальные символы строки **sFormatString** останутся без изменений. Если размер результирующей строки превышает 255 символов, то она будет обрезана до 255 символов.

Таблица 3.2.8 – Описание входов и выходов функции TIME_TO_STRING_FORMAT

Имя переменной	Тип	Описание
Входные переменные		
tTimeToConvert	TIME	Время
sFormatString	STRING(255)	Форматированная строка с заполнителем времени
Выходные переменные		
TIME_TO_STRING_FORMAT	STRING(255)	Форматированная строка с датой

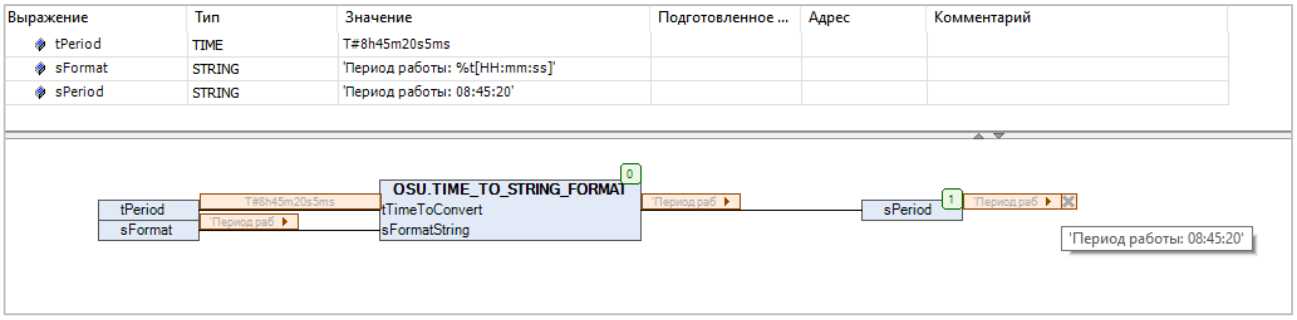


Рисунок 3.2.6 – Пример использования функции TIME_TO_STRING_FORMAT на языке CFC

3.3 Выделение подстрок

3.3.1 Функция Before

Функция **Before** возвращает фрагмент исходной строки **sSource**, предшествующий первому вхождению подстроки **sPostfix** (не включая саму подстроку). Все переменные функции имеют тип **STRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.1 – Описание входов и выходов функции Before

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPostfix	STRING(255)	Подстрока
Выходные переменные		
Before	STRING(255)	Фрагмент исходной строки, предшествующий первому вхождению подстроки (не включая саму подстроку)

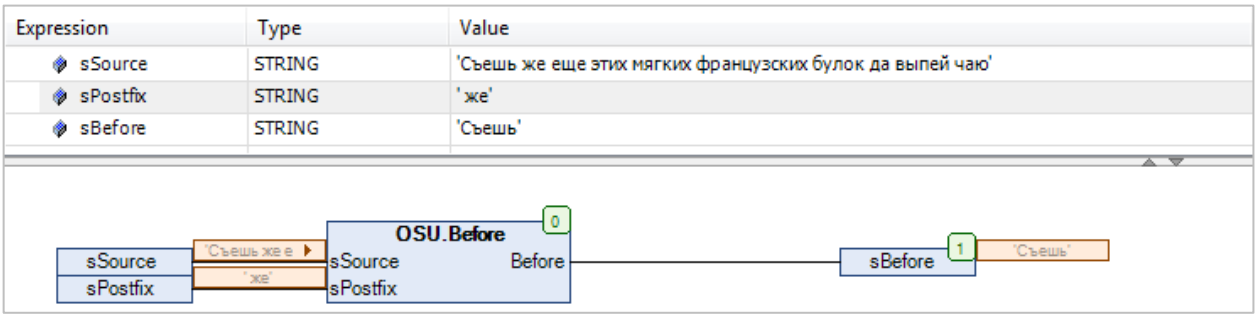


Рисунок 3.3.1 – Пример использования функции Before на языке CFC

3.3.2 Функция WBefore

Функция **WBefore** возвращает фрагмент исходной строки **wsSource**, предшествующий первому вхождению подстроки **wsPostfix** (не включая саму подстроку). Все переменные функции имеют тип **WSTRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.2 – Описание входов и выходов функции WBefore

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPostfix	WSTRING(255)	Подстрока
Выходные переменные		
WBefore	WSTRING(255)	Фрагмент исходной строки, предшествующий первому вхождению подстроки (не включая саму подстроку)

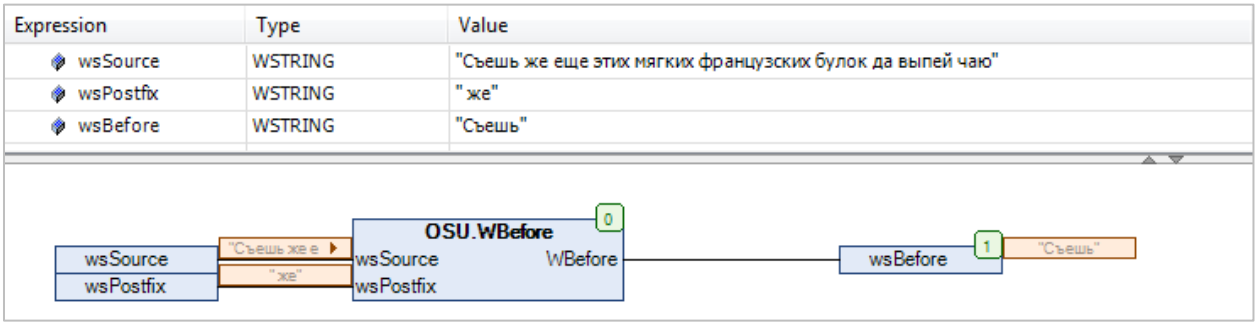


Рисунок 3.3.2 – Пример использования функции WBefore на языке CFC

3.3.3 Функция BeforeByNumber

Функция **BeforeByNumber** возвращает фрагмент исходной строки **sSource**, предшествующий n-му вхождению подстроки **sPostfix** (не включая саму подстроку), где **n** определяется значением входа **usiPostfixNumber**. Если **usiPostfixNumber** = **0**, то это значение интерпретируется как **1**. Вызов функции с **usiPostfixNumber** = **1** аналогичен вызову функции Before.

Все переменные функции имеют тип **STRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.3 – Описание входов и выходов функции BeforeByNumber

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPostfix	STRING(255)	Подстрока
usiPostfixNumber	STRING(255)	Номер вхождения подстроки
Выходные переменные		
BeforeByNumber	STRING(255)	Фрагмент исходной строки, предшествующий n-му вхождению подстроки (не включая саму подстроку)

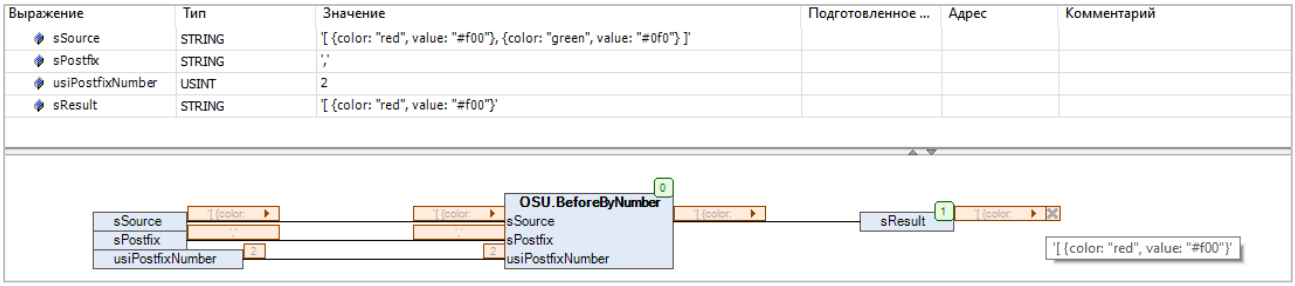


Рисунок 3.3.3 – Пример использования функции BeforeByNumber на языке CFC

3.3.4 Функция WBeforeByNumber

Функция **WBeforeBefore** возвращает фрагмент исходной строки **wsSource**, предшествующий n-му вхождению подстроки **wsPostfix** (не включая саму подстроку), где **n** определяется значением входа **usiPostfixNumber**. Если **usiPostfixNumber** = **0**, то это значение интерпретируется как **1**. Вызов функции с **usiPostfixNumber** = **1** аналогичен вызову функции WBefore.

Все переменные функции имеют тип **WSTRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.4 – Описание входов и выходов функции WBeforeByNumber

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPostfix	WSTRING(255)	Подстрока
usiPostfixNumber	WSTRING(255)	Номер вхождения подстроки
Выходные переменные		
WBeforeByNumber	WSTRING(255)	Фрагмент исходной строки, предшествующий n-му вхождению подстроки (не включая саму подстроку)

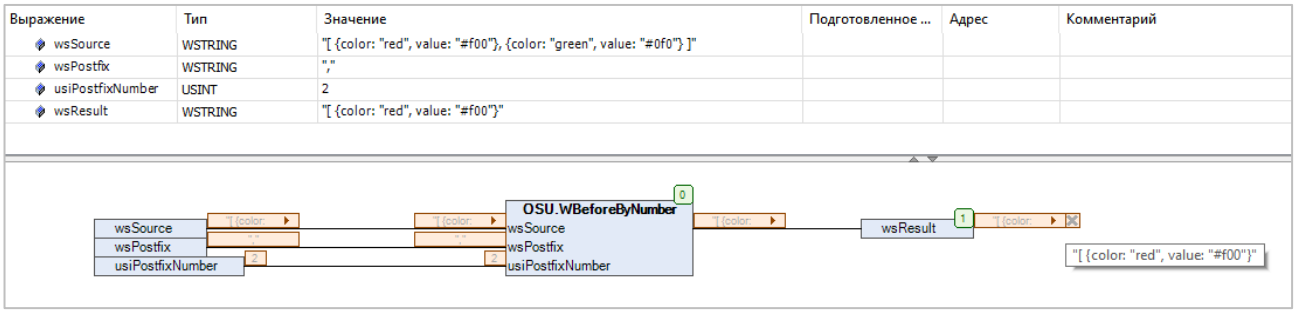


Рисунок 3.3.4 – Пример использования функции WBeforeByNumber на языке CFC

3.3.5 Функция After

Функция **After** возвращает фрагмент исходной строки **sSource**, следующий за первым вхождением подстроки **sPrefix** (не включая саму подстроку). Все переменные функции имеют тип **STRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.5 – Описание входов и выходов функции After

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPrefix	STRING(255)	Подстрока
Выходные переменные		
After	STRING(255)	Фрагмент исходной строки, следующий за первым вхождением подстроки (не включая саму подстроку)

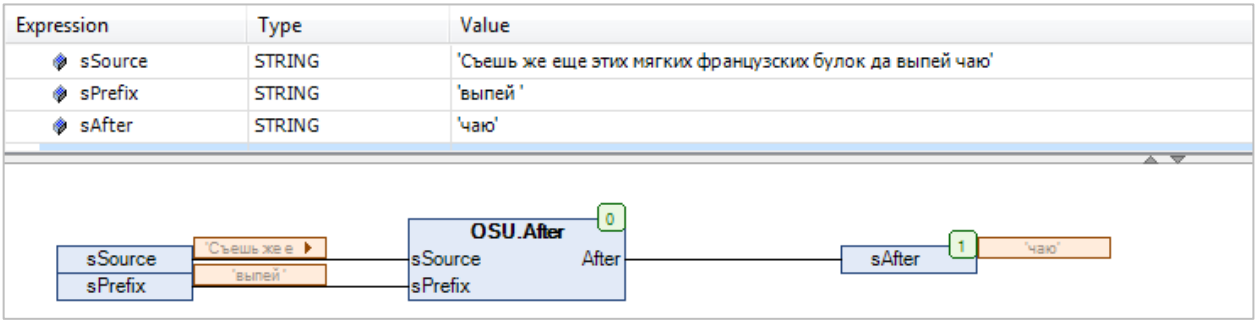


Рисунок 3.3.5 – Пример использования функции After на языке CFC

3.3.6 Функция WAfter

Функция **WAfter** возвращает фрагмент исходной строки **wsSource**, следующий за первым вхождением подстроки **wsPrefix** (не включая саму подстроку). Все переменные функции имеют тип **WSTRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.6 – Описание входов и выходов функции WAfter

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPrefix	WSTRING(255)	Подстрока
Выходные переменные		
WAfter	WSTRING(255)	Фрагмент исходной строки, следующий за первым вхождением подстроки (не включая саму подстроку)

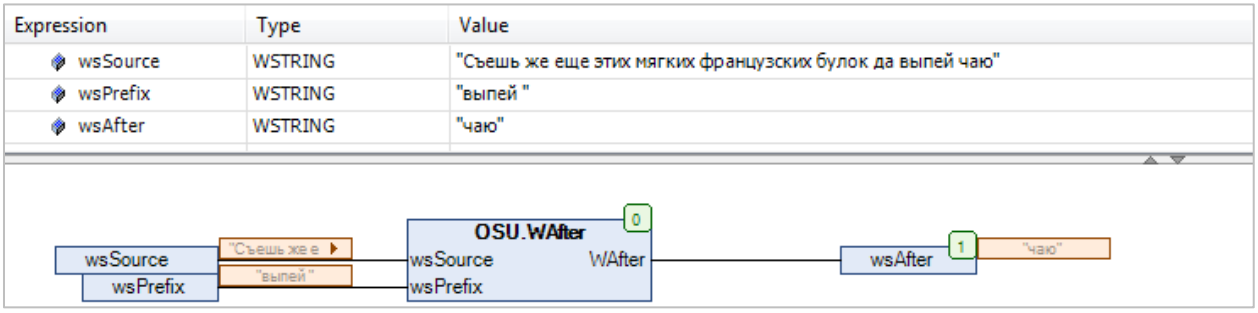


Рисунок 3.3.6 – Пример использования функции WAfter на языке CFC

3.3.7 Функция AfterByNumber

Функция **AfterByNumber** возвращает фрагмент исходной строки **sSource**, следующий за n-м вхождением подстроки **sPrefix** (не включая саму подстроку), где **n** определяется значением входа **usiPrefixNumber**. Если **usiPrefixNumber** = **0**, то это значение интерпретируется как **1**. Вызов функции с **usiPrefixNumber** = **1** аналогичен вызову функции After.

Все переменные функции имеют тип **STRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.7 – Описание входов и выходов функции AfterByNumber

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPrefix	STRING(255)	Подстрока
usiPrefixNumber	STRING(255)	Номер вхождения подстроки
Выходные переменные		
AfterByNumber	STRING(255)	Фрагмент исходной строки, следующий за n-м вхождением подстроки (не включая саму подстроку)

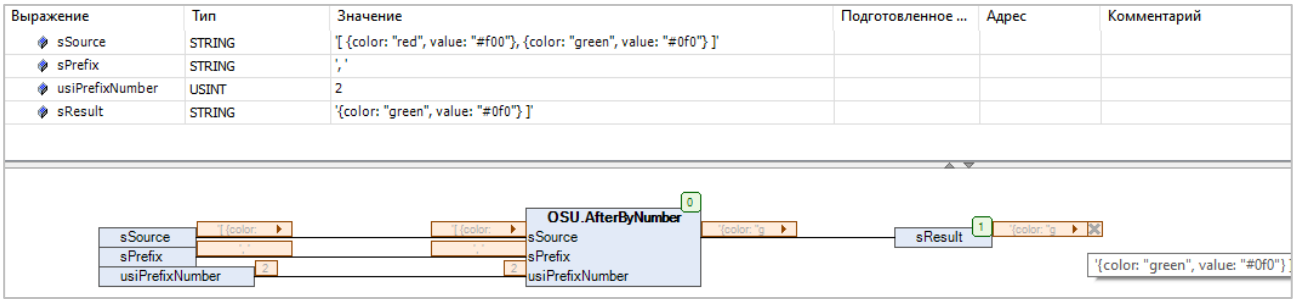


Рисунок 3.3.7 – Пример использования функции AfterByNumber на языке CFC

3.3.8 Функция WAfterByNumber

Функция **WAfterByNumber** возвращает фрагмент исходной строки **wsSource**, следующий за n-м вхождением подстроки **sPrefix** (не включая саму подстроку), где **n** определяется значением входа **usiPrefixNumber**. Если **usiPrefixNumber** = **0**, то это значение интерпретируется как **1**. Вызов функции с **usiPrefixNumber** = **1** аналогичен вызову функции WAfter.

Все переменные функции имеют тип **WSTRING**. Если подстрока не найдена, то функция возвращает пустую строку.

Таблица 3.3.8 – Описание входов и выходов функции WAfterByNumber

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPrefix	WSTRING(255)	Подстрока
usiPrefixNumber	WSTRING(255)	Номер вхождения подстроки
Выходные переменные		
WAfterByNumber	WSTRING(255)	Фрагмент исходной строки, следующий n-м вхождением подстроки (не включая саму подстроку)

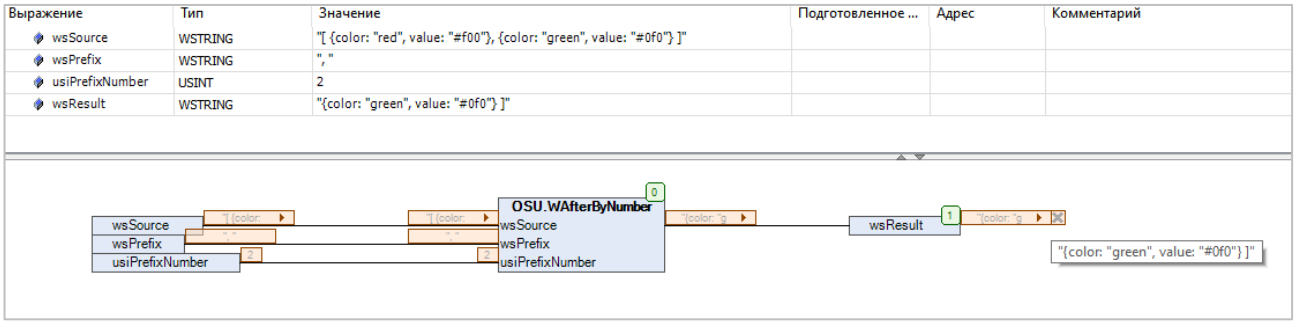


Рисунок 3.3.8 – Пример использования функции WAfterByNumber на языке CFC

3.3.9 Функция Between

Функция **Between** возвращает фрагмент исходной строки **sSource**, расположенный между первыми вхождениями начальной подстроки **sPrefix** и конечной подстроки **sPostfix** (не включая сами подстроки). Все переменные функции имеют тип **STRING**. Если префикс не задан, то поведение функции аналогично Before. Если постфикс не задан, то поведение функции аналогично After. Если префикс и/или постфикс не найдены в исходной строке или же постфикс найден перед префиксом – то функция вернет пустую строку. Если значения префикса и постфикса совпадают – то функция вернет фрагмент исходной строки, расположенный между первым и вторым вхождением данной подстроки.

Таблица 3.3.9 – Описание входов и выходов функции Between

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPrefix	STRING(255)	Начальная подстрока
sPostfix	STRING(255)	Конечная подстрока
Выходные переменные		
Between	STRING(255)	Фрагмент исходной строки, расположенный между первыми вхождениями начальной и конечной подстрок (не включая сами подстроки)

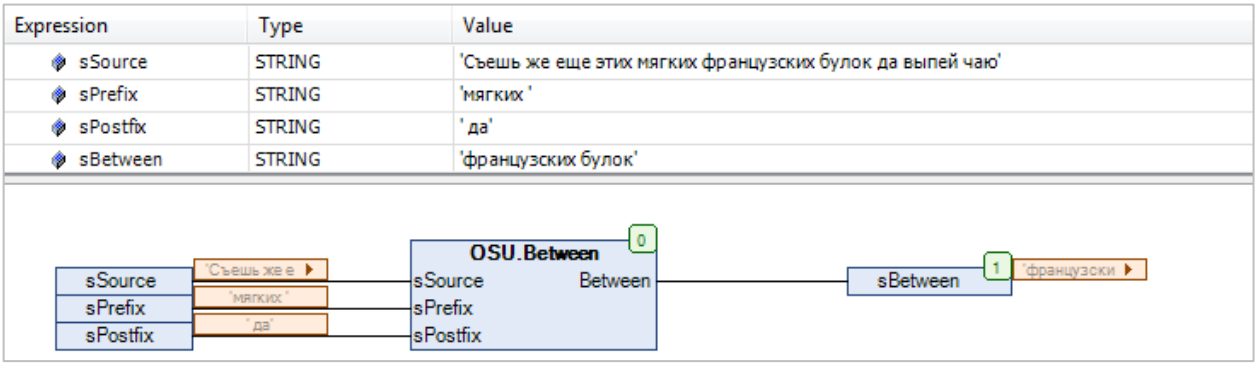


Рисунок 3.3.9 – Пример использования функции Between на языке CFC

3.3.10 Функция WBetween

Функция **WBetween** возвращает фрагмент исходной строки **wsSource**, расположенный между первыми вхождениями начальной подстроки **wsPrefix** и конечной подстроки **wsPostfix** (не включая сами подстроки). Все переменные функции имеют тип **WSTRING**. Если префикс не задан, то поведение функции аналогично WBefore. Если постфикс не задан, то поведение функции аналогично WAfter. Если префикс и/или постфикс не найдены в исходной строке или же постфикс найден перед префиксом – то функция вернет пустую строку. Если значения префикса и постфикса совпадают – то функция вернет фрагмент исходной строки, расположенный между первым и вторым вхождением данной подстроки.

Таблица 3.3.10 – Описание входов и выходов функции WBetween

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPrefix	WSTRING(255)	Начальная подстрока
wsPostfix	WSTRING(255)	Конечная подстрока
Выходные переменные		
WBetween	WSTRING(255)	Фрагмент исходной строки, расположенный между первыми вхождениями начальной и конечной подстрок (не включая сами подстроки)

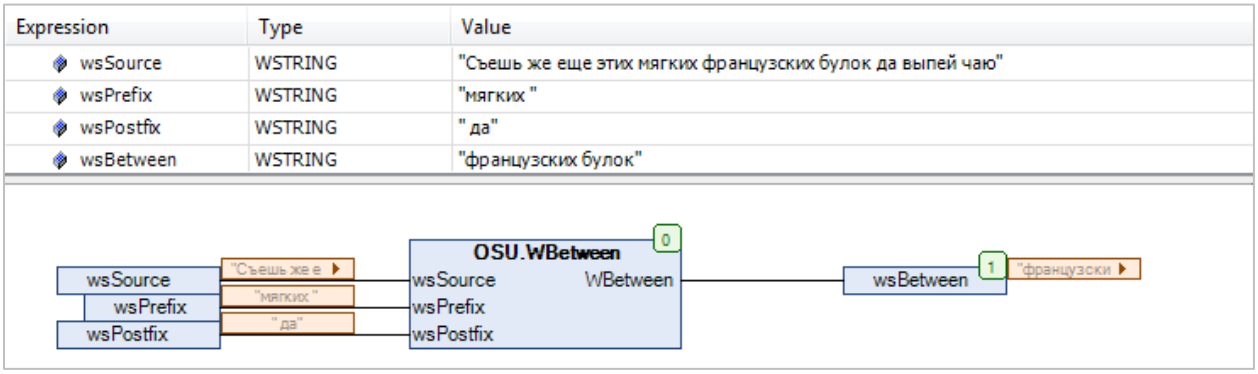


Рисунок 3.3.10 – Пример использования функции WBetween на языке CFC

3.3.11 Функция BetweenByNumber

Функция **BetweenByNumber** возвращает фрагмент исходной строки **sSource**, расположенный между начальной подстрокой **sPrefix** и конечной подстрокой **sPostfix** (не включая сами подстроки). При этом:

- если вход **xPostfix** имеет значение **TRUE**, то в исходной строке ищется n-е (определяемое значением входа **usiNumber**) значение постфикса и «ближайшее» к нему значение префикса;
- если вход **xPostfix** имеет значение **FALSE**, то в исходной строке ищется n-е (определяемое значением входа **usiNumber**) значение префикса и «ближайшее» к нему значение постфикса.

Все строковые переменные функции имеют тип **STRING**. Если **usiNumber** = **0**, то это значение интерпретируется как **1.** Значения префикса и постфикса могут совпадать. Если префикс и/или постфикс не найдены в исходной строке – то функция вернет пустую строку. Если префикс не задан, то поведение функции аналогично Before. Если постфикс не задан, то поведение функции аналогично After.

Таблица 3.3.11 – Описание входов и выходов функции BetweenByNumber

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPrefix	STRING(255)	Начальная подстрока
sPostfix	STRING(255)	Конечная подстрока
usiNumber	USINT	Номер подстроки (тип подстроки определяется входом **xPostfix**)
xPostfix	BOOL	**TRUE** – искать n-е вхождение постфикса и «ближайшее» к нему значение префикса, **FALSE** – искать n-е вхождение префикса и «ближайшее» к нему значение постфикса
Выходные переменные		
BetweenByNumber	STRING(255)	Фрагмент исходной строки, расположенный между префиксом и постфиксом

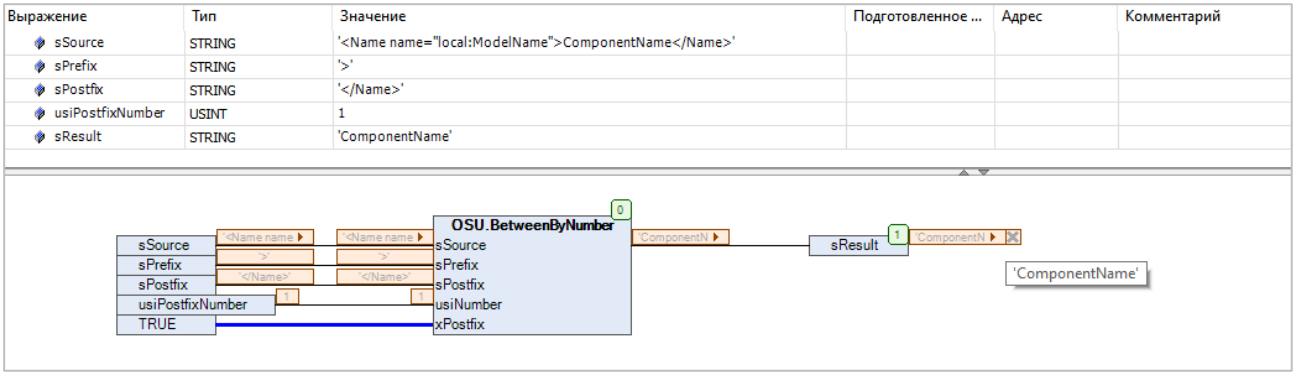


Рисунок 3.3.11 – Пример использования функции BetweenByNumber на языке CFC с xPostfix = TRUE

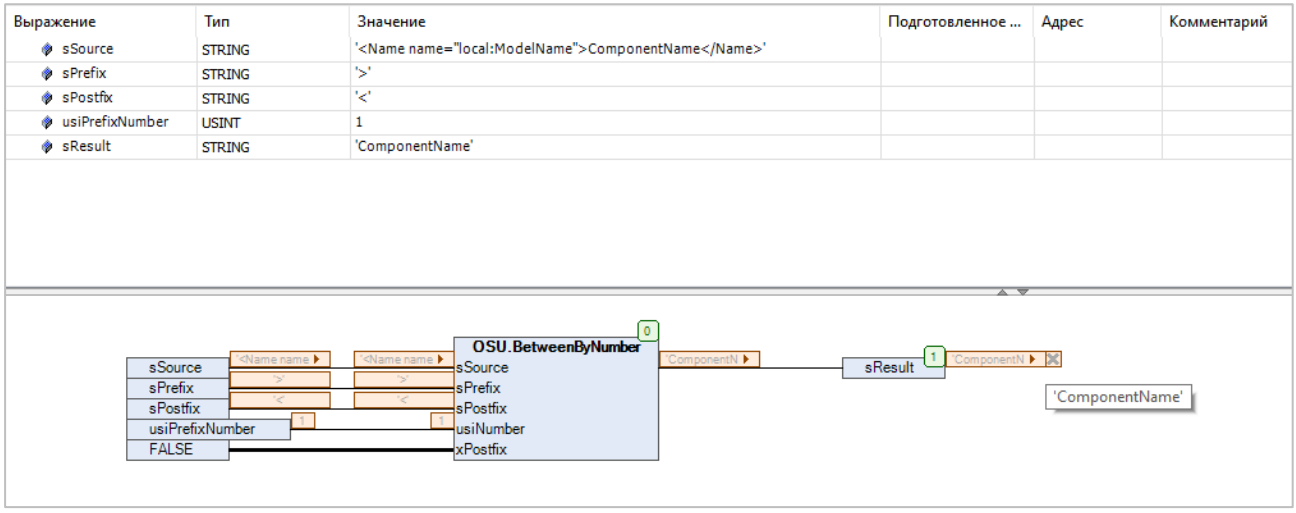


Рисунок 3.3.12 – Пример использования функции BetweenByNumber на языке CFC с xPostfix = FALSE

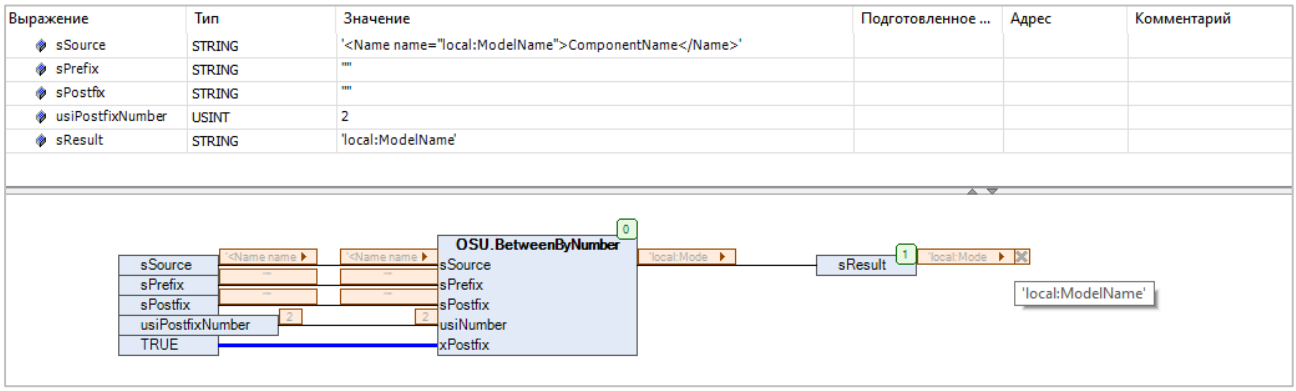


Рисунок 3.3.13 – Пример использования функции BetweenByNumber на языке CFC с xPostfix = TRUE и sPrefix = sPostfix

3.3.12 Функция WBetweenByNumber

Функция **WBetweenByNumber** возвращает фрагмент исходной строки **wsSource**, расположенный между начальной подстрокой **wsPrefix** и конечной подстрокой **wsPostfix** (не включая сами подстроки). При этом:

- если вход **xPostfix** имеет значение **TRUE**, то в исходной строке ищется n-е (определяемое значение входа **usiNumber**) значение постфикса и «ближайшее» к нему значение префикса;
- если вход **xPostfix** имеет значение **FALSE**, то в исходной строке ищется n-е (определяемое значение входа **usiNumber**) значение префикса и «ближайшее» к нему значение постфикса.

Все строковые переменные функции имеют тип **WSTRING**. Если **usiNumber** = **0**, то это значение интерпретируется как **1.** Значения префикса и постфикса могут совпадать. Если префикс и/или постфикс не найдены в исходной строке – то функция вернет пустую строку. Если префикс не задан, то поведение функции аналогично WBefore. Если постфикс не задан, то поведение функции аналогично WAfter.

Таблица 3.3.12 – Описание входов и выходов функции WBetweenByNumber

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPrefix	WSTRING(255)	Начальная подстрока
wsPostfix	WSTRING(255)	Конечная подстрока
usiNumber	USINT	Номер подстроки (тип подстроки определяется входом **xPostfix**)
xPostfix	BOOL	**TRUE** – искать n-е вхождение постфикса и «ближайшее» к нему значение префикса, **FALSE** – искать n-е вхождение префикса и «ближайшее» к нему значение постфикса
Выходные переменные		
WBetweenByNumber	WSTRING(255)	Фрагмент исходной строки, расположенный между префиксом и постфиксом

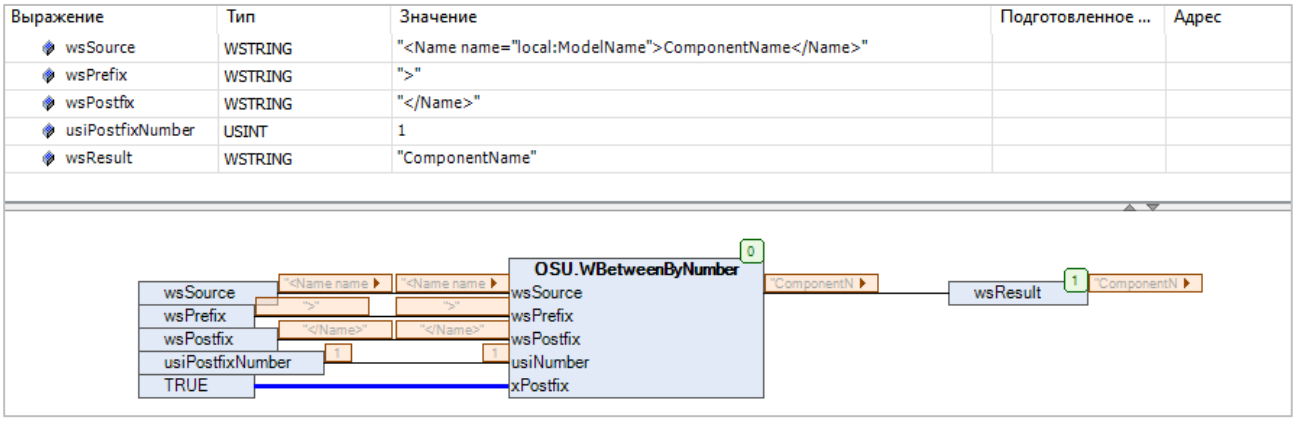


Рисунок 3.3.14 – Пример использования функции WBetweenByNumber на языке CFC с xPostfix = TRUE

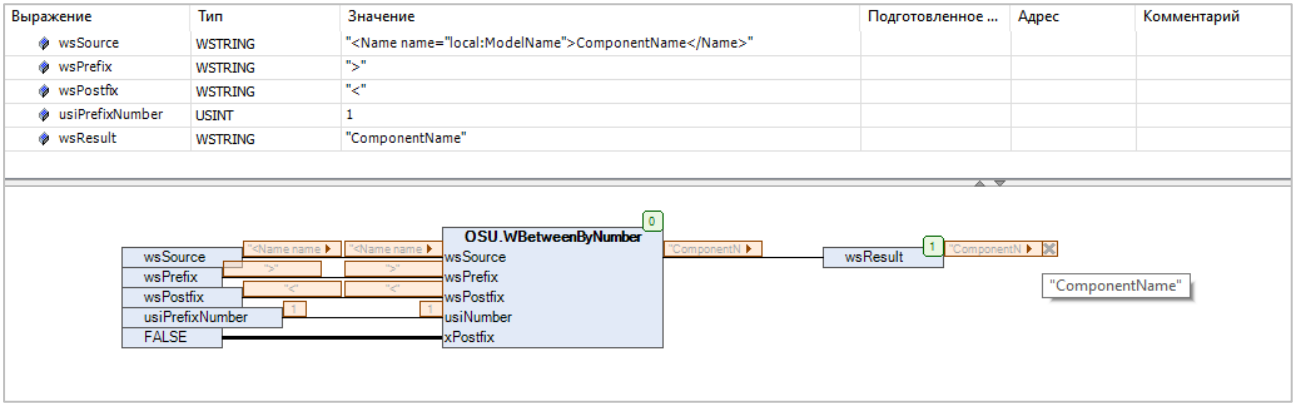


Рисунок 3.3.15 – Пример использования функции WBetweenByNumber на языке CFC с xPostfix = FALSE

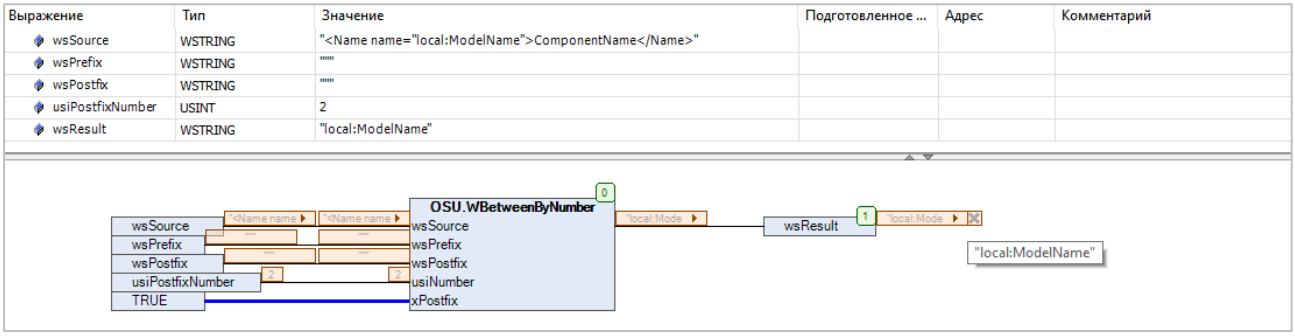


Рисунок 3.3.16 – Пример использования функции WBetweenByNumber на языке CFC с xPostfix = TRUE и sPrefix = sPostfix

3.3.13 Функция BetweenByNumber2

Функция **BetweenByNumber2** возвращает фрагмент исходной строки **sSource**, расположенный между n-м (определяемым значением входа **usiPrefixNumber**) вхождением начальной подстроки **sPrefix** и m-м (определяемым значением входа **usiPostfixNumber**) конечной подстроки **sPostfix** (не включая сами подстроки). Все строковые переменные функции имеют тип **STRING**. Если **usiPrefixNumber или usiPostfixNumber** = **0**, то это значение интерпретируется как **1.** Значения префикса и постфикса могут совпадать. Если префикс и/или постфикс не найдены в исходной строке или же постфикс найден перед префиксом – то функция вернет пустую строку.

Таблица 3.3.13 – Описание входов и выходов функции BetweenByNumber2

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sPrefix	STRING(255)	Начальная подстрока
sPostfix	STRING(255)	Конечная подстрока
usiPrefixNumber	USINT	Номер начальной подстроки
usiPostfixNumber	USINT	Номер конечной подстроки
Выходные переменные		
BetweenByNumber2	STRING(255)	Фрагмент исходной строки, расположенный между префиксом и постфиксом

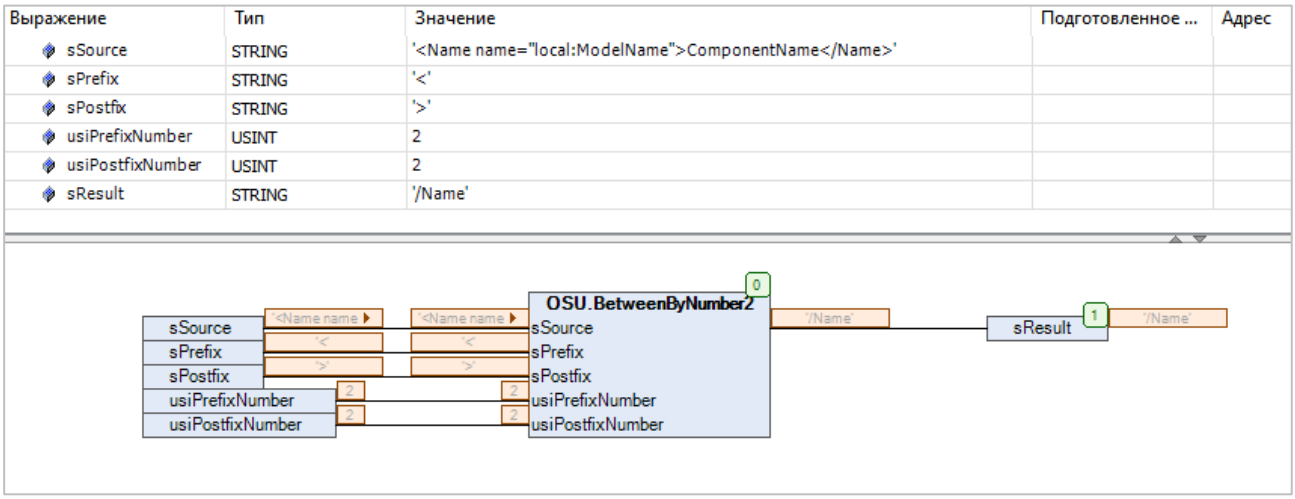


Рисунок 3.3.17 – Пример использования функции BetweenByNumber2 на языке CFC

3.3.14 Функция WBetweenByNumber2

Функция **WBetweenByNumber2** возвращает фрагмент исходной строки **wsSource**, расположенный между n-м (определяемым значением входа **usiPrefixNumber**) вхождением начальной подстроки **wsPrefix** и m-м (определяемым значением входа **usiPostfixNumber**) конечной подстроки **wsPostfix** (не включая сами подстроки). Все строковые переменные функции имеют тип **WSTRING**. Если **usiPrefixNumber или usiPostfixNumber** = **0**, то это значение интерпретируется как **1.** Значения префикса и постфикса могут совпадать. Если префикс и/или постфикс не найдены в исходной строке или же постфикс найден перед префиксом – то функция вернет пустую строку.

Таблица 3.3.14 – Описание входов и выходов функции WBetweenByNumber2

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsPrefix	WSTRING(255)	Начальная подстрока
wsPostfix	WSTRING(255)	Конечная подстрока
usiPrefixNumber	USINT	Номер начальной подстроки
usiPostfixNumber	USINT	Номер конечной подстроки
Выходные переменные		
WBetweenByNumber2	WSTRING(255)	Фрагмент исходной строки, расположенный между префиксом и постфиксом

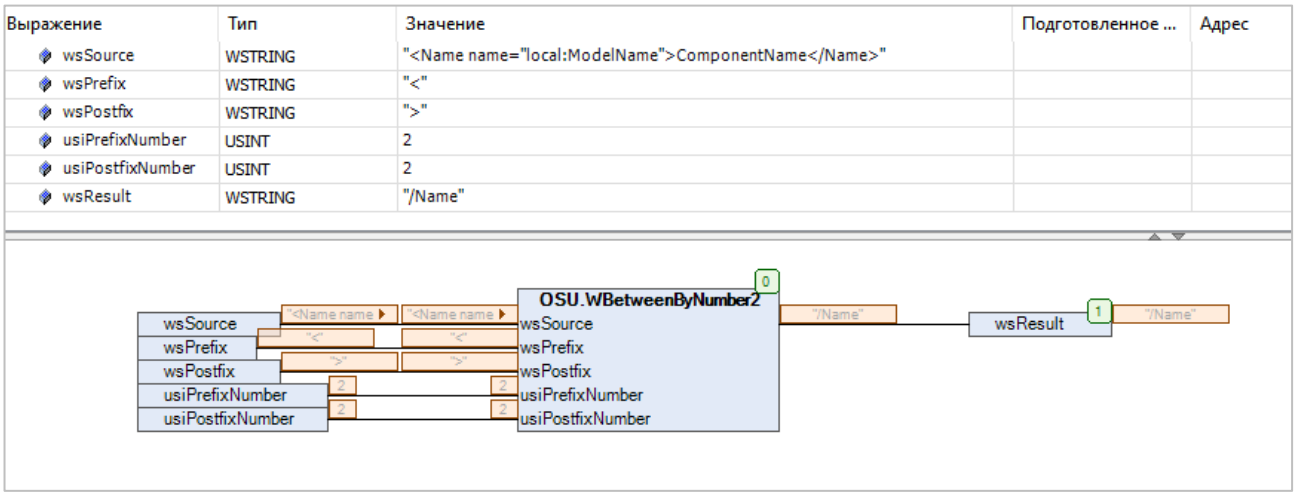


Рисунок 3.3.18 – Пример использования функции WBetweenByNumber2 на языке CFC

3.3.15 Функция SplitStringByToken

Функция **SplitStringByToken** разделяет исходную строку, размещенную по указателю **psSourceString** и имеющую длину **uiSourceLen** (длина может превышать 255 символов), на подстроки, которые будут размещены в массиве по указателю **pasSubstrings**. Число элементов массива определяется значением входа **usiMaxSubstringCount**, а размер подстроки в байтах – значением входа **uiSubstringSize**. В качестве токена (строки-разделителя) используется значение входа **sToken**. Все строковые переменные функции имеют тип **STRING**. Функция возвращает число разделенных ею подстрок или же **-1** (признак ошибки входных данных), если выполняется хотя бы одно из следующих условий:

- psSourceString = 0;
- uiSourceLen = 0;
- sToken = '';
- pasSubstrings = 0;
- uiSubstringSize = 0;
- usiMaxSubstringsCount = 0.

Таблица 3.3.15 – Описание входов и выходов функции SplitStringByToken

Имя переменной	Тип	Описание
Входные переменные		
psSourceString	PONTER TO BYTE	Указатель на исходную строку (**STRING**)
uiSourceLen	UINT	Длина исходной строки
sToken	STRING(255)	Токен, по которому будут разделены подстроки
pasSubstrings	POINTER TO BYTE	Указатель на массив, в котором будут размещены подстроки
uiSubstringSize	UINT	Размер одной подстроки массива в байтах
usiMaxSubstringsCount	USINT	Число элементов массива
Выходные переменные		
SplitStringByToken	INT	Число подстрок, записанных в массив, или **-1**, если хотя бы один из входов имеет некорректное значение

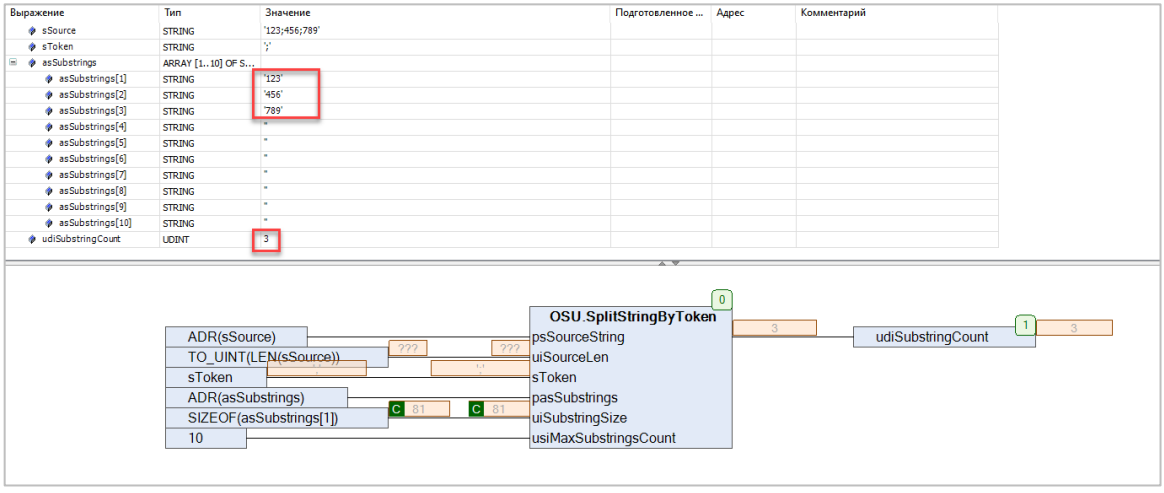


Рисунок 3.3.19 – Пример использования функции SplitStringByToken на языке CFC

3.3.16 Функция WSplitStringByToken

Функция **WSplitStringByToken** разделяет исходную строку, размещенную по указателю **pwsSourceString** и имеющую длину **uiSourceLen** (длина может превышать 255 символов), на подстроки, которые будут размещены в массиве по указателю **pawsSubstrings**. Число элементов массива определяется значением входа **usiMaxSubstringCount**, а размер подстроки в байтах – значением входа **uiSubstringSize**. В качестве токена (строки-разделителя) используется значение входа **wsToken**. Все строковые переменные функции имеют тип **WSTRING**. Функция возвращает число разделенных ею подстрок или же **-1** (признак ошибки входных данных), если выполняется хотя бы одно из следующих условий:

- pwsSourceString = 0;
- uiSourceLen = 0;
- wsToken = "";
- pawsSubstrings = 0;
- uiSubstringSize = 0;
- usiMaxSubstringsCount = 0.

Таблица 3.3.16 – Описание входов и выходов функции WSplitStringByToken

Имя переменной	Тип	Описание
Входные переменные		
pwsSourceString	PONTER TO WORD	Указатель на исходную строку (**WSTRING**)
uiSourceLen	UINT	Длина исходной строки
wsToken	WSTRING(255)	Токен, по которому будут разделены подстроки
pawsSubstrings	POINTER TO WORD	Указатель на массив, в котором будут размещены подстроки
uiSubstringSize	UINT	Размер одной подстроки массива в байтах
usiMaxSubstringsCount	USINT	Число элементов массива
Выходные переменные		
WSplitStringByToken	INT	Число подстрок, записанных в массив, или **-1**, если хотя бы один из входов имеет некорректное значение

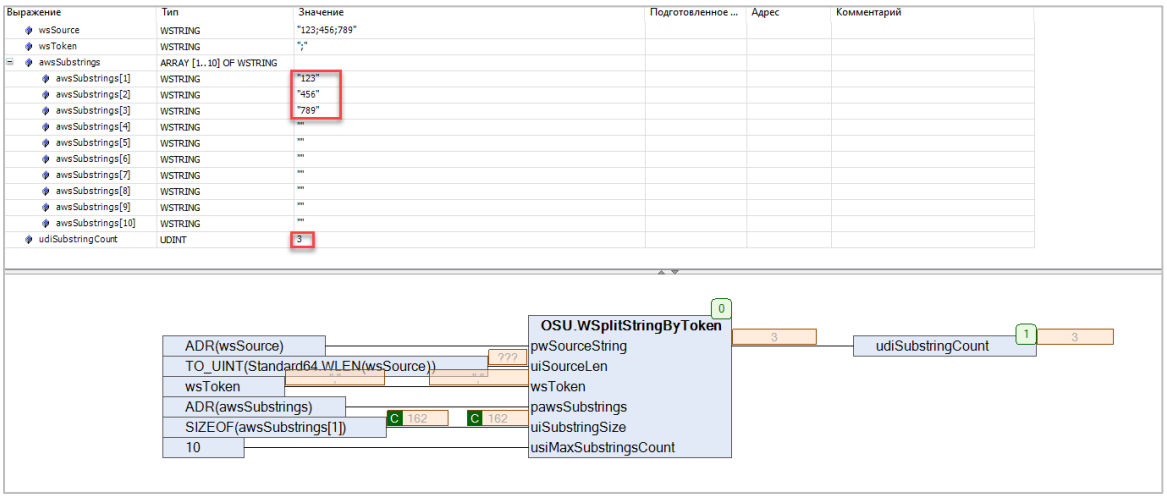


Рисунок 3.3.20 – Пример использования функции WSplitStringByToken на языке CFC

3.4 Дополнение строк

3.4.1 Функция ADD_CHAR

Функция **ADD_CHAR** дополняет строку типа STRING **sInputString** символом **sAddChar** до длины **usiTargetLen** справа (при **xRight** := **TRUE**) или слева (при **xRight** := **FALSE**).
Если длина **sInputString** > **usiTargetLen**, то функция возвращает **sInputString** без преобразований.

Таблица 3.4.1 – Описание входов и выходов функции ADD_CHAR

Имя переменной	Тип	Описание
Входные переменные		
sInputString	STRING(255)	Исходная строка
usiTargetLen	USINT	Длина результирующей строки
sAddChar	STRING(1)	Символ-заполнитель
xRight	BOOL	Режим дополнения строки (**TRUE** – справа, **FALSE** – слева)
Выходные переменные		
ADD_CHAR	STRING(255)	Строка, дополненная символами-заполнителями

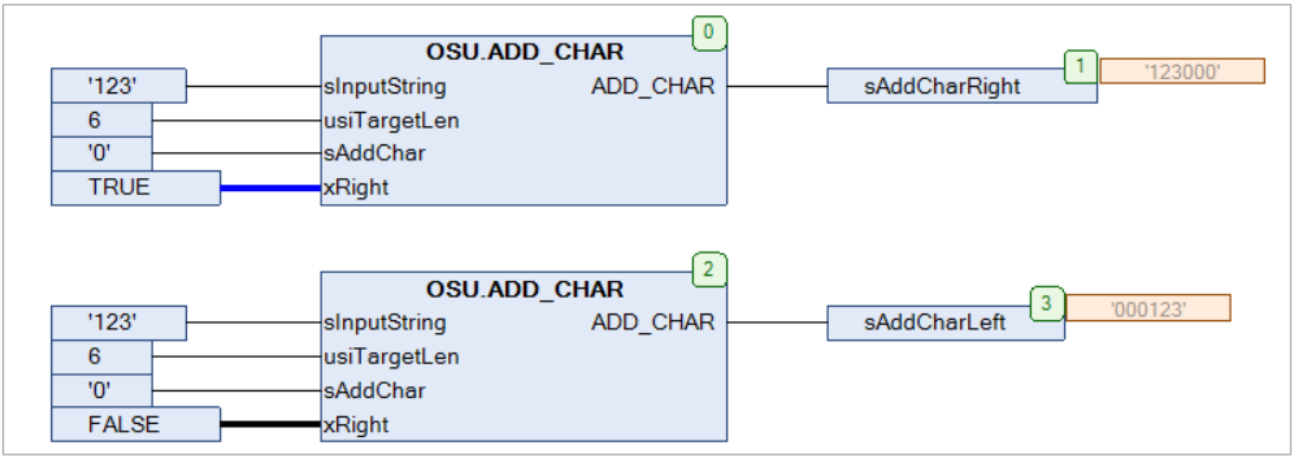


Рисунок 3.4.1 – Пример использования функции ADD_CHAR на языке CFC

3.4.2 Функция WADD_CHAR

Функция **WADD_CHAR** дополняет строку типа WSTRING **wsInputString** символом **wsAddChar** до длины **usiTargetLen** справа (при **xRight** := **TRUE**) или слева (при **xRight** := **FALSE**).
Если длина **wsInputString** > **usiTargetLen**, то функция возвращает **wsInputString** без преобразований.

Таблица 3.4.2 – Описание входов и выходов функции WADD_CHAR

Имя переменной	Тип	Описание
Входные переменные		
wsInputString	WSTRING(255)	Исходная строка
usiTargetLen	USINT	Длина результирующей строки
wsAddChar	WSTRING(1)	Символ-заполнитель
xRight	BOOL	Режим дополнения строки (**TRUE** – справа, **FALSE** – слева)
Выходные переменные		
WADD_CHAR	WSTRING(255)	Строка, дополненная символами-заполнителями

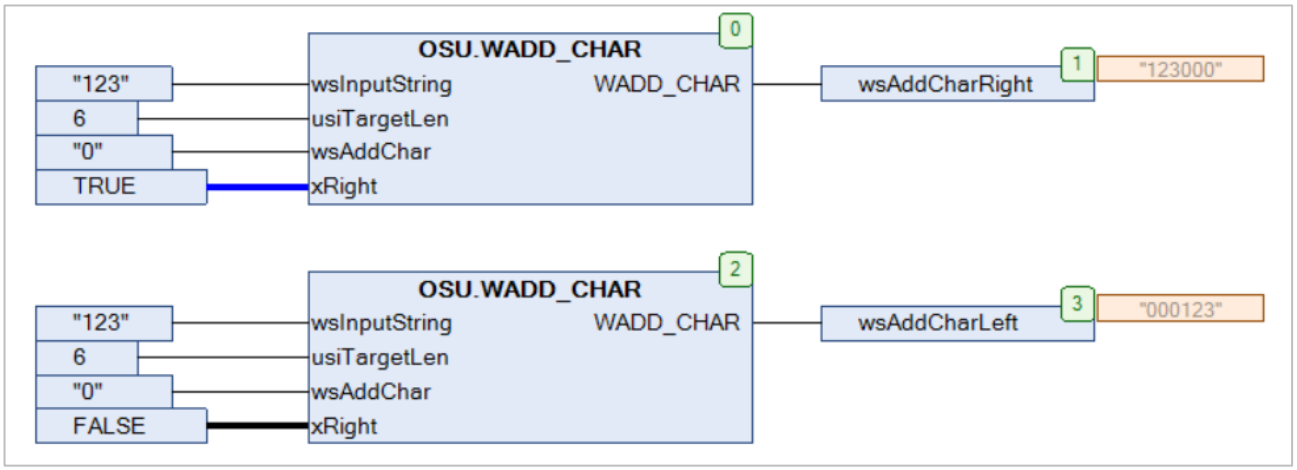


Рисунок 3.4.2 – Пример использования функции WADD_CHAR на языке CFC

3.5 Замена подстрок

3.5.1 Функция ReplaceSubstring

Функция **ReplaceSubstring** заменяет первое вхождение искомой подстроки **sWhatToReplace** в исходной строке **sSource** на подстроку **sReplaceWith**. Если искомая подстрока не найдена, то функция возвращает исходную строку. Все переменные функции имеют тип **STRING**.

Таблица 3.5.1 – Описание входов и выходов функции ReplaceSubstring

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sWhatToReplace	STRING(255)	Искомая подстрока
sReplaceWith	STRING(255)	Замещающая подстрока
Выходные переменные		
ReplaceSubstring	STRING(255)	Строка с замещенной подстрокой

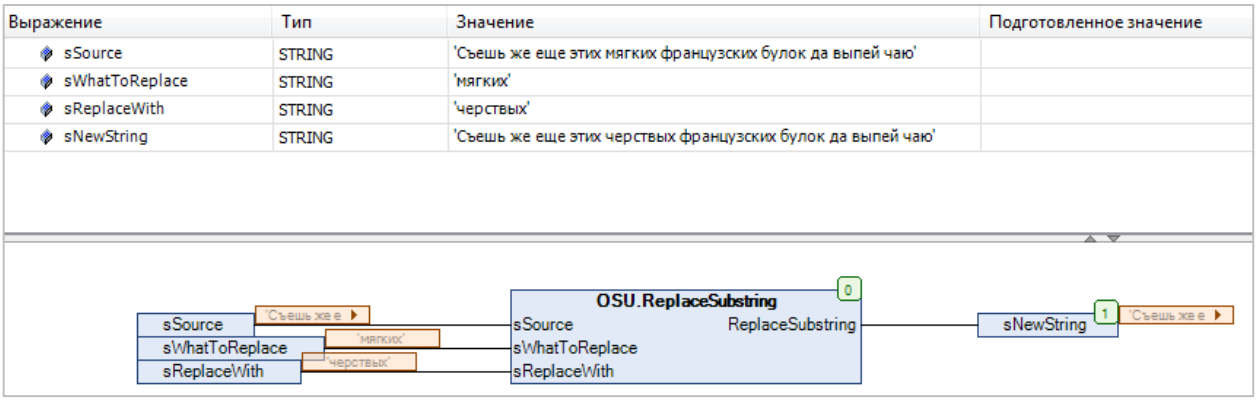


Рисунок 3.5.1 – Пример использования функции ReplaceSubstring на языке CFC

3.5.2 Функция WReplaceSubstring

Функция **WReplaceSubstring** заменяет первое вхождение искомой подстроки **wsWhatToReplace** в исходной строке **wsSource** на подстроку **wsReplaceWith**. Если искомая подстрока не найдена, то функция возвращает исходную строку. Все переменные функции имеют тип **WSTRING**.

Таблица 3.5.2 – Описание входов и выходов функции WReplaceSubstring

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsWhatToReplace	WSTRING(255)	Искомая подстрока
wsReplaceWith	WSTRING(255)	Замещающая подстрока
Выходные переменные		
WReplaceSubstring	WSTRING(255)	Строка с замещенной подстрокой

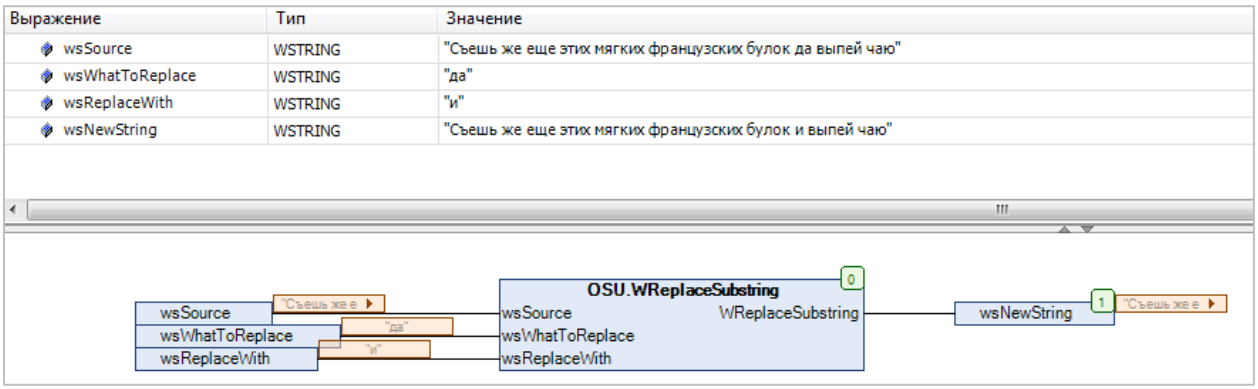


Рисунок 3.5.2 – Пример использования функции WReplaceSubstring на языке CFC

3.5.3 Функция ReplaceAllSubstrings

Функция **ReplaceAllSubstrings** заменяет все вхождения искомой подстроки **sWhatToReplace** в исходной строке **sSource** на подстроку **sReplaceWith**. Если искомая подстрока не найдена, то функция возвращает исходную строку. Все переменные функции имеют тип **STRING**.

Таблица 3.5.3 – Описание входов и выходов функции ReplaceAllSubstrings

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sWhatToReplace	STRING(255)	Искомая подстрока
sReplaceWith	STRING(255)	Замещающая подстрока
Выходные переменные		
ReplaceAllSubstrings	STRING(255)	Строка с замещенными подстроками

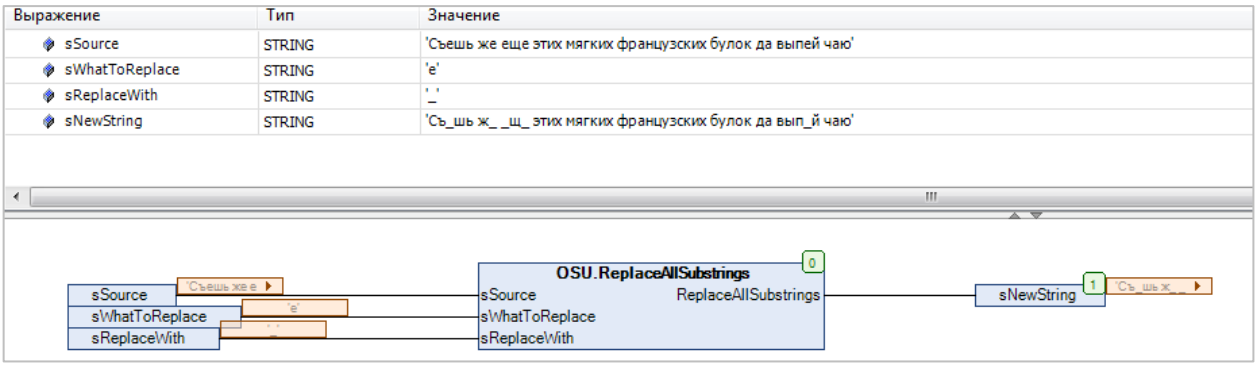


Рисунок 3.5.3 – Пример использования функции ReplaceAllSubstrings на языке CFC

3.5.4 Функция WReplaceAllSubstrings

Функция **WReplaceAllSubstrings** заменяет все вхождения искомой подстроки **wsWhatToReplace** в исходной строке **wsSource** на подстроку **wsReplaceWith**. Если искомая подстрока не найдена, то функция возвращает исходную строку. Все переменные функции имеют тип **WSTRING**.

Таблица 3.5.4 – Описание входов и выходов функции WReplaceAllSubstrings

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsWhatToReplace	WSTRING(255)	Искомая подстрока
wsReplaceWith	WSTRING(255)	Замещающая подстрока
Выходные переменные		
WReplaceAllSubstrings	WSTRING(255)	Строка с замещенными подстроками

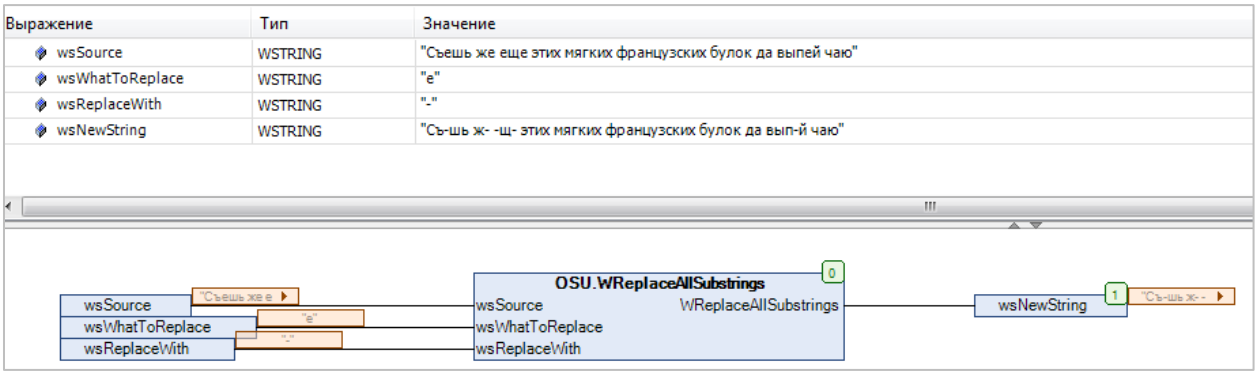


Рисунок 3.5.4 – Пример использования функции WReplaceAllSubstrings на языке CFC

3.6 Конкатенация строк

3.6.1 Функция CONCAT4

Функция **CONCAT4** объединяет входные строки типа STRING **sInputString1…4** в одну строку.

Таблица 3.6.1 – Описание входов и выходов функции CONCAT4

Имя переменной	Тип	Описание
Входные переменные		
sInputString1…4	STRING(255)	Исходные строки
Выходные переменные		
CONCAT4	STRING(255)	Объединенная строка

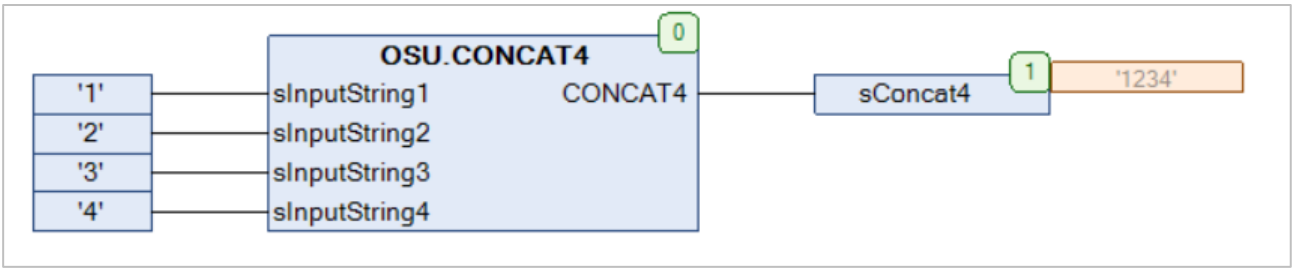


Рисунок 3.6.1 – Пример использования функции CONCAT4 на языке CFC

3.6.2 Функция WCONCAT4

Функция **WCONCAT4** объединяет входные строки типа WSTRING **wsInputString1…4** в одну строку.

Таблица 3.6.2 – Описание входов и выходов функции WCONCAT4

Имя переменной	Тип	Описание
Входные переменные		
wsInputString1…4	WSTRING(255)	Исходные строки
Выходные переменные		
WCONCAT4	WSTRING(255)	Объединенная строка

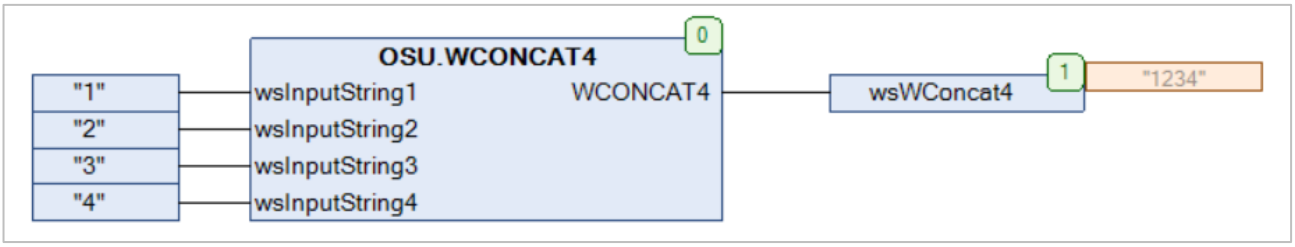


Рисунок 3.6.2 – Пример использования функции WCONCAT4 на языке CFC

3.6.3 Функция CONCAT8

Функция **CONCAT8** объединяет входные строки типа STRING **sInputString1…8** в одну строку.

Таблица 3.6.3 – Описание входов и выходов функции CONCAT8

Имя переменной	Тип	Описание
Входные переменные		
sInputString1…8	STRING(255)	Исходные строки
Выходные переменные		
CONCAT8	STRING(255)	Объединенная строка

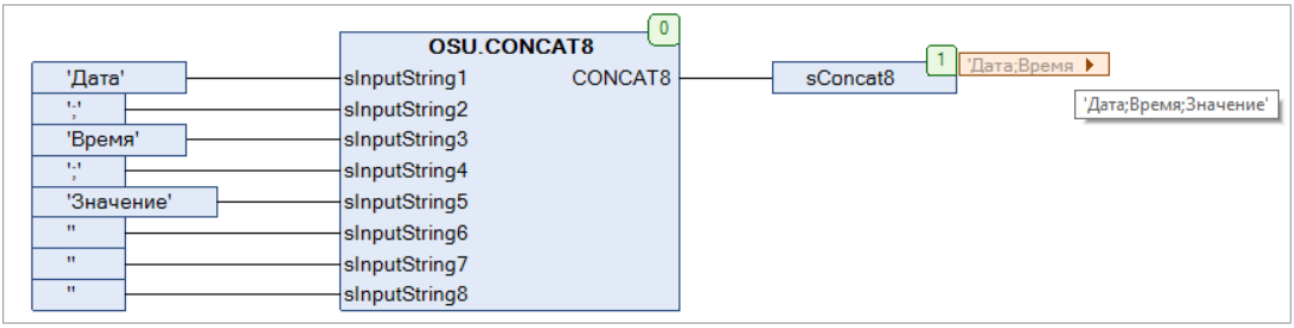


Рисунок 3.6.3 – Пример использования функции CONCAT8 на языке CFC

3.6.4 Функция WCONCAT8

Функция **WCONCAT8** объединяет входные строки типа WSTRING **wsInputString1…8** в одну строку.

Таблица 3.6.4 – Описание входов и выходов функции WCONCAT8

Имя переменной	Тип	Описание
Входные переменные		
wsInputString1…8	WSTRING(255)	Исходные строки
Выходные переменные		
WCONCAT8	WSTRING(255)	Объединенная строка

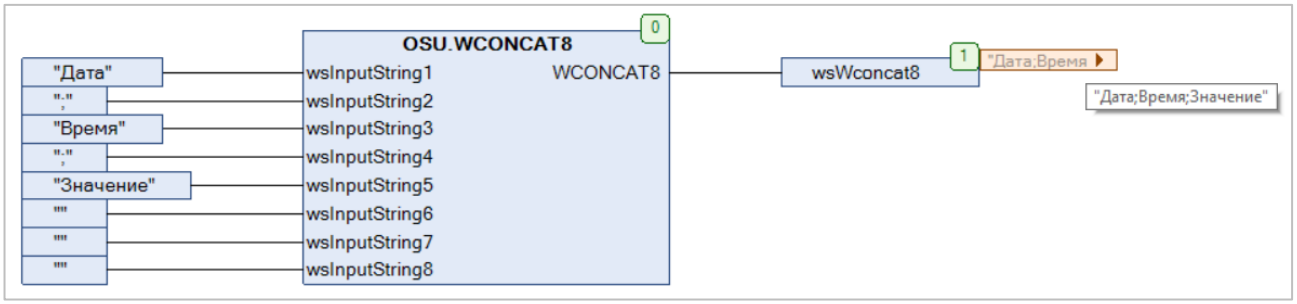


Рисунок 3.6.4 – Пример использования функции WCONCAT8 на языке CFC

3.7 Поиск подстрок

3.7.1 Функция FindSubstringPosAfterN

Функция **FindSubstringPosAfterN** возвращает позицию первого вхождения искомой подстроки **sWhatToFind** в исходную строку **sSource**. Начальная позиция для поиска определяется входом **uiSearchFrom**. Если искомая подстрока не найдена, то функция возвращает **0**. Строковые переменные функции имеют тип **STRING**.

Таблица 3.7.1 – Описание входов и выходов функции FindSubstringPosAfterN

Имя переменной	Тип	Описание
Входные переменные		
sSource	STRING(255)	Исходная строка
sWhatToFind	STRING(255)	Искомая подстрока
uiSearchFrom	UINT	Начальная позиция для поиска
Выходные переменные		
FindSubstringPosAfterN	UINT	Позиция вхождения искомой подстроки в исходную строку

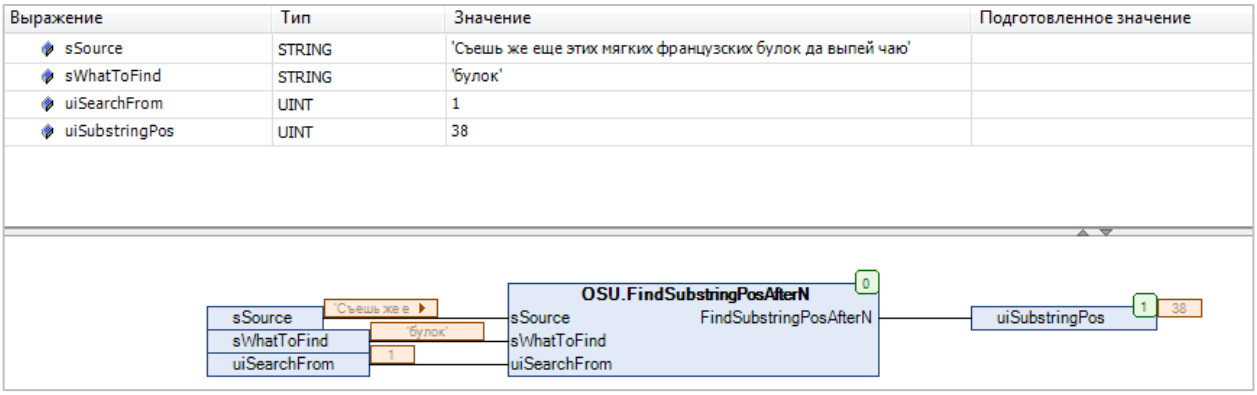


Рисунок 3.7.1 – Пример использования функции FindSubstringPosAfterN на языке CFC

3.7.2 Функция WFindSubstringPosAfterN

Функция **WFindSubstringPosAfterN** возвращает позицию первого вхождения искомой подстроки **wsWhatToFind** в исходную строку **wsSource**. Начальная позиция для поиска определяется входом **uiSearchFrom**. Если искомая подстрока не найдена, то функция возвращает **0**. Строковые переменные функции имеют тип **WSTRING**.

Таблица 3.7.2 – Описание входов и выходов функции WFindSubstringPosAfterN

Имя переменной	Тип	Описание
Входные переменные		
wsSource	WSTRING(255)	Исходная строка
wsWhatToFind	WSTRING(255)	Искомая подстрока
uiSearchFrom	UINT	Начальная позиция для поиска
Выходные переменные		
WFindSubstringPosAfterN	UINT	Позиция вхождения искомой подстроки в исходную строку

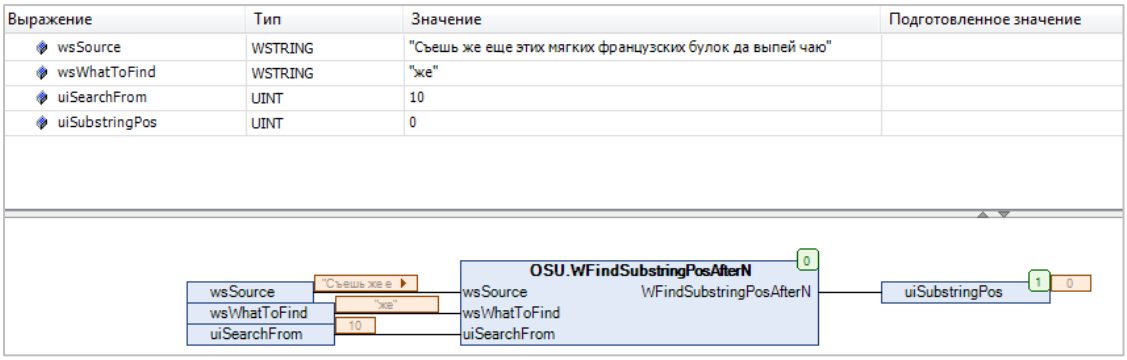


Рисунок 3.7.2 – Пример использования функции WFindSubstringPosAfterN на языке CFC

3.8 Преобразование IP и MAC

3.8.1 Функция BYTES_TO_IPSTRING

Функция **BYTES_TO_IPSTRING** возвращает строковое представление IP-адреса, заданного в виде массива байт **abyIpAddr**.

Таблица 3.8.1 – Описание входов и выходов функции BYTES_TO_IPSTRING

Имя переменной	Тип	Описание
Входные переменные		
abyIpAddr	ARRAY [0..3] OF BYTE	IP-адрес в виде массива байт
Выходные переменные		
BYTES_TO_IPSTRING	STRING(15)	IP-адрес в строковом виде

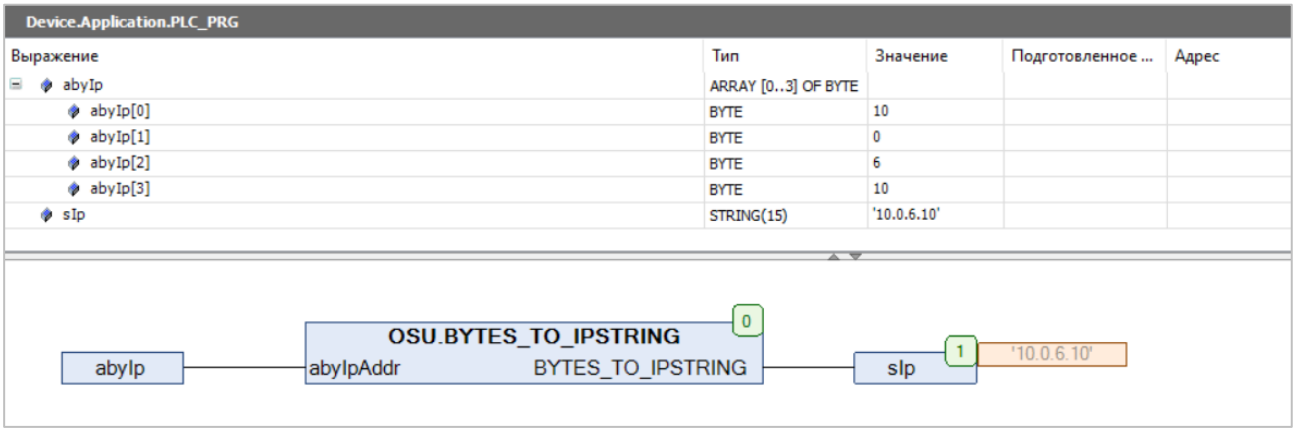


Рисунок 3.8.1 – Пример использования функции BYTES_TO_IPSTRING на языке CFC

3.8.2 Функция IPSTRING_TO_BYTES

Функция **IPSTRING_TO_BYTES** конвертирует строковое представление IP-адреса **sIpAddr** в массив байт. Вход **sIpAddr** не валидируется.

Таблица 3.8.2 – Описание входов и выходов функции IPSTRING_TO_BYTES

Имя переменной	Тип	Описание
Входные переменные		
sIpAddr	STRING(15)	IP-адрес в строковом виде
Выходные переменные		
IPSTRING_TO_BYTES	ARRAY [0..3] OF BYTE	IP-адрес в виде массива байт

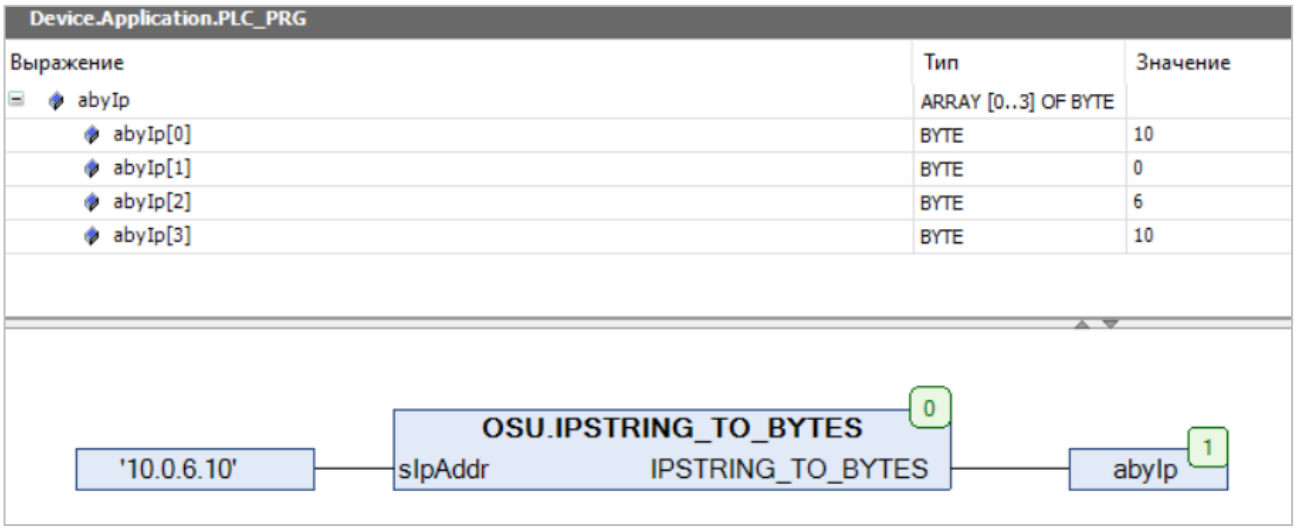


Рисунок 3.8.2 – Пример использования функции IPSTRING_TO_BYTES на языке CFC

3.8.3 Функция UDINT_TO_IPSTRING

Функция **UDINT_TO_IPSTRING** возвращает строковое представление IP-адреса, заданного в виде переменной типа UDINT **udiIpAddr**.

Таблица 3.8.3 – Описание входов и выходов функции UDINT_TO_IPSTRING

Имя переменной	Тип	Описание
Входные переменные		
udiIpAddr	UDINT	IP-адрес в бинарном виде
Выходные переменные		
UDINT_TO_IPSTRING	STRING	IP-адрес в строковом виде

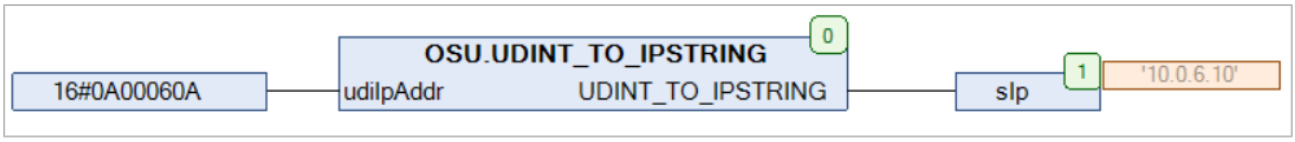


Рисунок 3.8.3 – Пример использования функции UDINT_TO_IPSTRING на языке CFC

3.8.4 Функция IPSTRING_TO_UDINT

Функция **IPSTRING_TO_UDINT** конвертирует строковое представление IP-адреса **sIpAddr** в бинарный вид. Вход **sIpAddr** не валидируется.

Таблица 3.8.4 – Описание входов и выходов функции IPSTRING_TO_UDINT

Имя переменной	Тип	Описание
Входные переменные		
sIpAddr	STRING	IP-адрес в строковом виде
Выходные переменные		
IPSTRING_TO_UDINT	UDINT	IP-адрес в бинарном виде

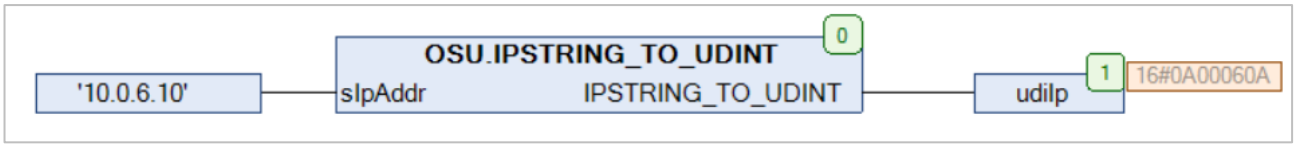


Рисунок 3.8.4 – Пример использования функции IPSTRING_TO_UDINT на языке CFC

3.8.5 Функция MAC_TO_STRING

Функция **MAC_TO_STRING** возвращает строковое представление MAC-адреса, заданного в виде массива байт **abyMacAddr**.

Таблица 3.8.5 – Описание входов и выходов функции MAC_TO_STRING

Имя переменной	Тип	Описание
Входные переменные		
abyMacAddr	ARRAY [0..5] OF BYTE	MAC-адрес в виде массива байт
Выходные переменные		
MAC_TO_STRING	STRING(17)	MAC-адрес в строковом виде

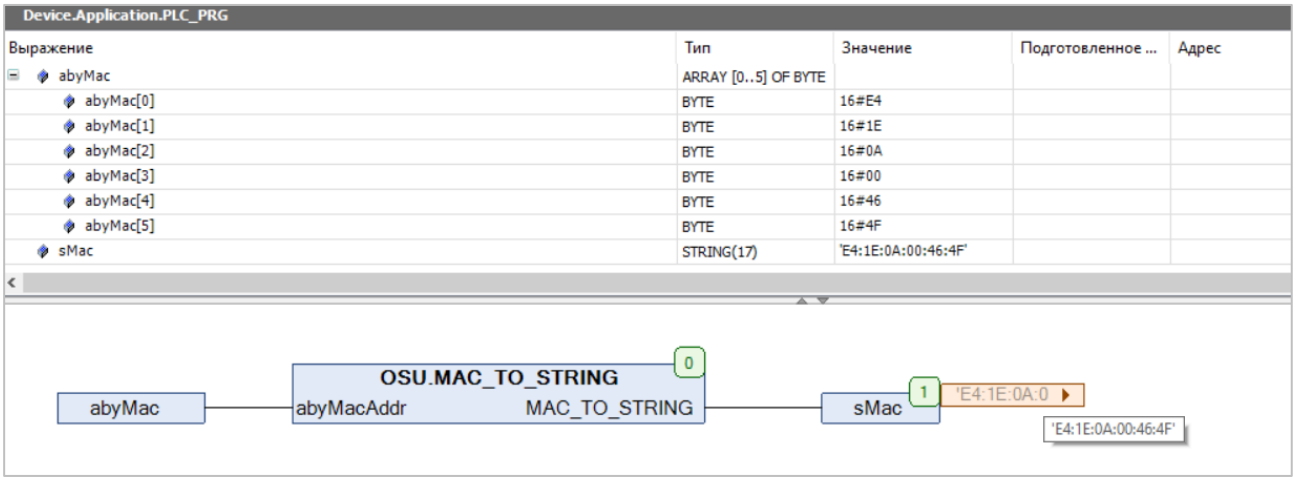


Рисунок 3.8.5 – Пример использования функции MAC_TO_STRING на языке CFC

3.9 Преобразование HEX-строк

3.9.1 Функция HEX_STR_TO_WORD

Функция **HEX_STR_TO_WORD** конвертирует строку с HEX-значением **sInputString** и префиксом **sPrefix** в переменную типа **WORD**, содержащую это значение в целочисленном виде. Исходная строка **sInputString** может включать в себя до 4 символов префикса и до 4 символов значения (от 0 до FFFF). Регистр HEX-символов не учитывается. Исходная строка не валидируется.

Если в исходной строке префикс отсутствует, то значение входа **sPrefix** следует оставить пустым.

Таблица 3.9.1 – Описание входов и выходов функции HEX_STR_TO_WORD

Имя переменной	Тип	Описание
Входные переменные		
sInputString	STRING(8)	Строка с HEX-значением
sPrefix	STRING(4)	Префикс исходной строки
Выходные переменные		
HEX_STR_TO_WORD	WORD	Результат преобразования

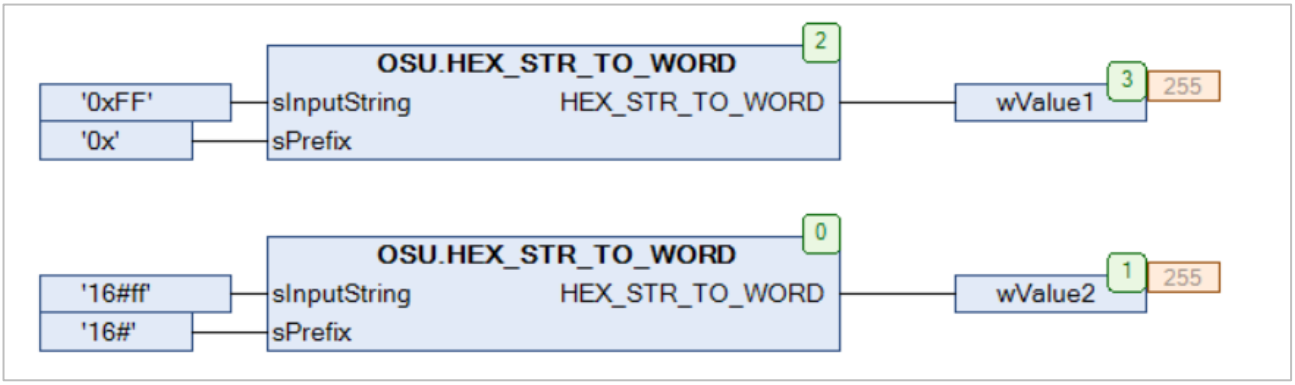


Рисунок 3.9.1 – Пример использования функции HEX_STR_TO_WORD на языке CFC

3.9.2 Функция WORD_TO_HEX_STR

Функция **WORD_TO_HEX_STR** конвертирует целочисленное значение типа WORD **wInput** в строку с его HEX-представлением и префиксом **sPrefix**.Если вход **xUpperCase** имеет значение **TRUE**, то HEX-символы строки имеют верхний регистр, если **FALSE** – то нижний.

Таблица 3.9.2 – Описание входов и выходов функции WORD_TO_HEX_STR

Имя переменной	Тип	Описание
Входные переменные		
wInput	WORD	Исходное значение
xUpperCase	BOOL	Регистр HEX-символов (**TRUE** – верхний, **FALSE** – нижний)
sPrefix	STRING(4)	Префикс формируемой строки
Выходные переменные		
WORD_TO_HEX_STR	STRING(8)	Строка с префиксом и HEX-значением

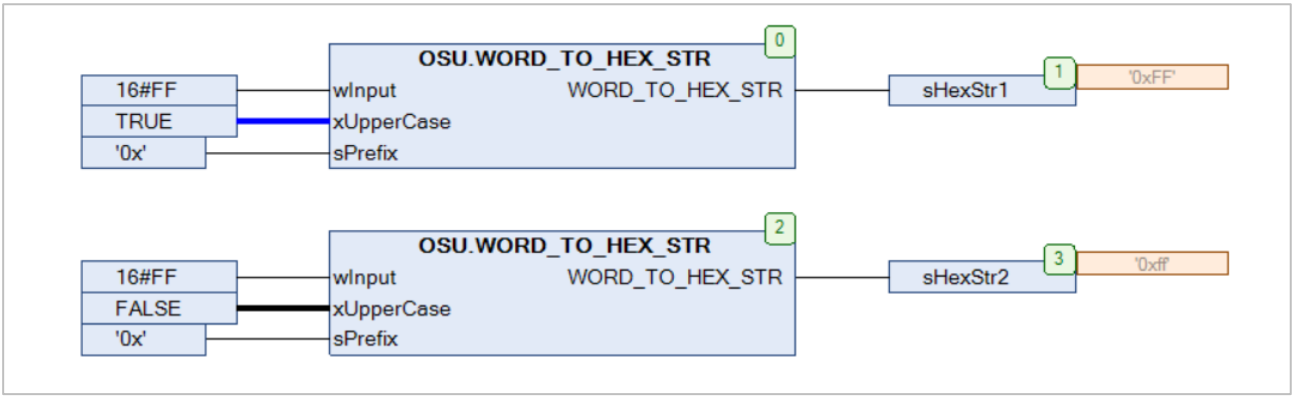


Рисунок 3.9.2 – Пример использования функции WORD_TO_HEX_STR на языке CFC

3.10 Преобразование регистра символов

3.10.1 Функция LowerCase

Функция **LowerCase** преобразует все символы исходной строки **sStringToConvert** (в кодировке CP1251) в нижний регистр. Все переменные функции имеют тип **STRING**.

Таблица 3.10.1 – Описание входов и выходов функции LowerCase

Имя переменной	Тип	Описание
Входные переменные		
sStringToConvert	STRING(255)	Исходная строка
Выходные переменные		
LowerCase	STRING(255)	Строка в нижнем регистре

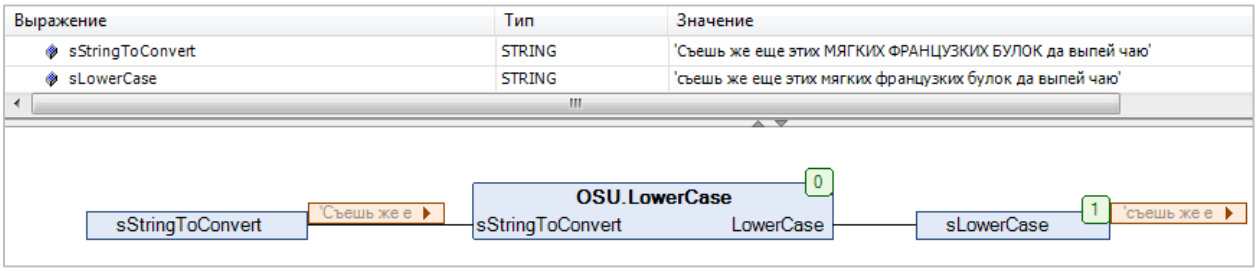


Рисунок 3.10.1 – Пример использования функции LowerCase на языке CFC

3.10.2 Функция WLowerCase

Функция **WLowerCase** преобразует символы русского и английского алфавита исходной строки **wsStringToConvert** в нижний регистр. Все переменные функции имеют тип **WSTRING**.

Таблица 3.10.2 – Описание входов и выходов функции WLowerCase

Имя переменной	Тип	Описание
Входные переменные		
wsStringToConvert	WSTRING(255)	Исходная строка
Выходные переменные		
WLowerCase	WSTRING(255)	Строка в нижнем регистре

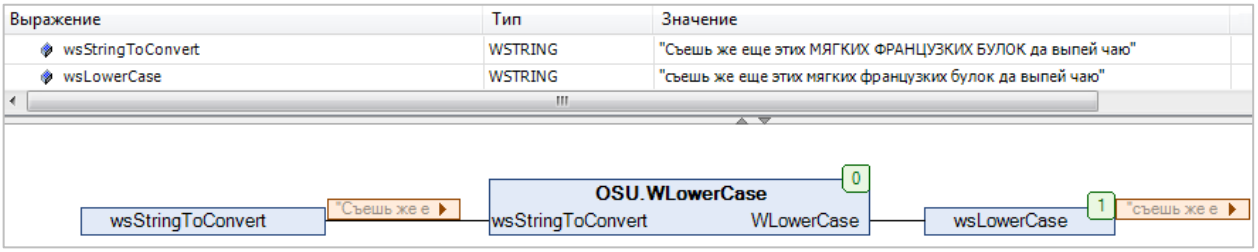


Рисунок 3.10.2 – Пример использования функции WLowerCase на языке CFC

3.10.3 Функция UpperCase

Функция **UpperCase** преобразует все символы исходной строки **sStringToConvert** (в кодировке CP1251) в верхний регистр. Все переменные функции имеют тип **STRING**.

Таблица 3.10.3 – Описание входов и выходов функции UpperCase

Имя переменной	Тип	Описание
Входные переменные		
sStringToConvert	STRING(255)	Исходная строка
Выходные переменные		
UpperCase	STRING(255)	Строка в верхнем регистре

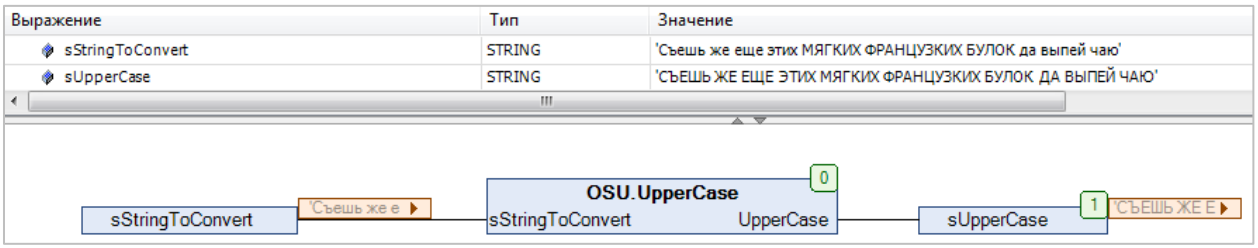


Рисунок 3.10.3 – Пример использования функции UpperCase на языке CFC

3.10.4 Функция WUpperCase

Функция **WUpperCase** преобразует символы русского и английского алфавита исходной строки **wsStringToConvert** в верхний регистр. Все переменные функции имеют тип **WSTRING**.

Таблица 3.10.4 – Описание входов и выходов функции WUpperCase

Имя переменной	Тип	Описание
Входные переменные		
wsStringToConvert	WSTRING(255)	Исходная строка
Выходные переменные		
WUpperCase	WSTRING(255)	Строка в верхнем регистре

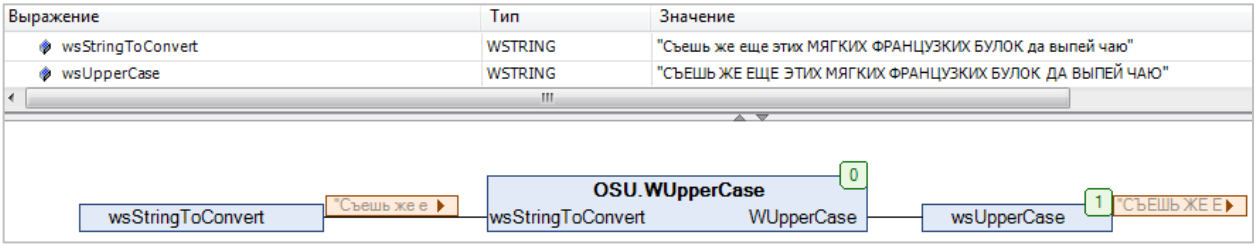


Рисунок 3.10.4 – Пример использования функции WUpperCase на языке CFC

3.11 Остальные функции

3.11.1 Функция GetCharType

Функция **GetCharType** возвращает тип символа с ASCII-кодом **byChar** (кодировка CP1251). Типы символов определены в перечислении **CHAR_TYPE**.

Таблица 3.11.1 – Описание входов и выходов функции GetCharType

Имя переменной	Тип	Описание
Входные переменные		
byChar	BYTE	ASCII-код тип символа
Выходные переменные		
GetCharType	CHAR_TYPE	Тип символа

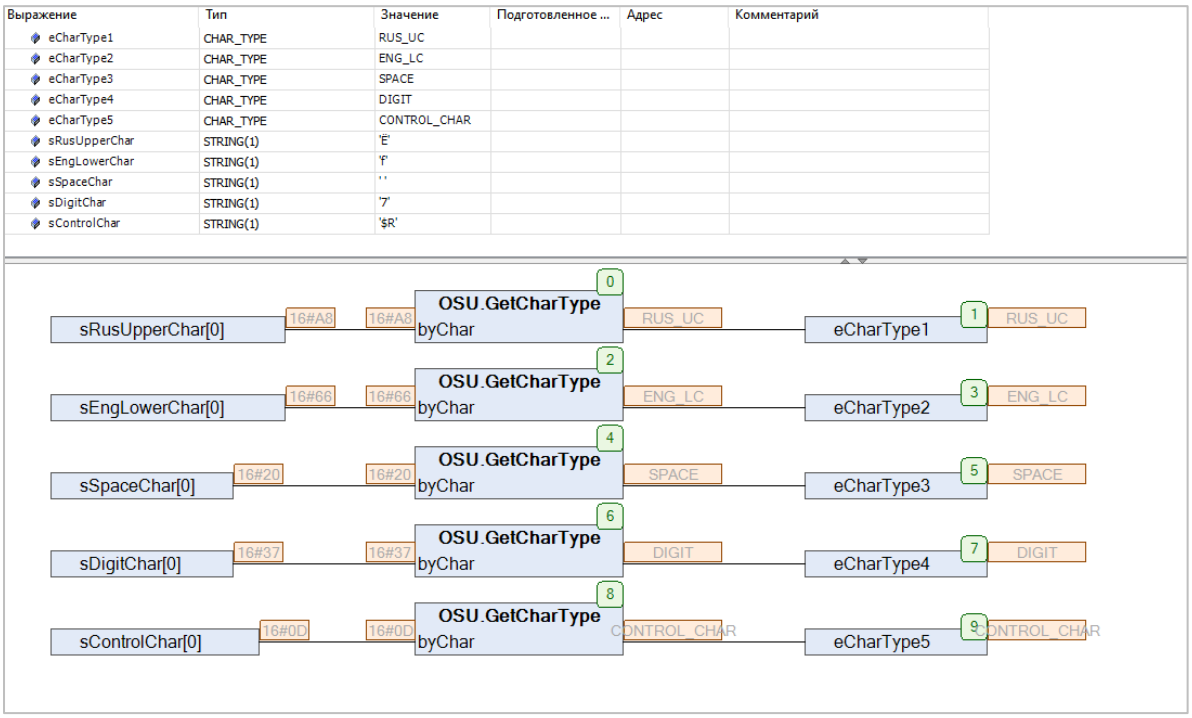


Рисунок 3.11.1 – Пример использования функции GetCharType на языке CFC

Таблица 3.11.2 – Описание элементов перечисления CHAR_TYPE

Название	Тип	Значение	Описание
RUS_LC	UINT	0	Символ кириллицы в нижнем регистре
RUS_UC	UINT	1	Символ кириллицы в верхнем регистре
ENG_LC	UINT	2	Символ латиницы в нижнем регистре
ENG_UC	UINT	3	Символ латиницы в верхнем регистре
DIGIT	UINT	4	Цифра
SPACE	UINT	5	Пробел
CONTROL_CHAR	UINT	6	Управляющий символ
OTHER	UINT	7	Другой символ (!, #, @ и т. д.)

3.11.2 Функция GetPathToDevice

Функция **GetPathToDevice** возвращает файловый заместитель для заданной директории. Директории определены в перечислении **DEVICE_DIR**. Файловые заместители могут использоваться в вызове POU из библиотек **CAA File**, **SysFile** и т. д., а также при использовании действия визуализации **Передача файла**. В системе исполнения они будут интерпретироваться как пути к соответствующим директориям.

Таблица 3.11.3 – Описание входов и выходов функции GetPathToDevice

Имя переменной	Тип	Описание
Входные переменные		
eDeviceDir	DEVICE_DIR	Директория контроллера
xAddLastSlash	BOOL	**TRUE** – к файловому заместителю будет добавлен символ '/'
Выходные переменные		
GetPathToDevice	STRING	Файловый заместитель директории

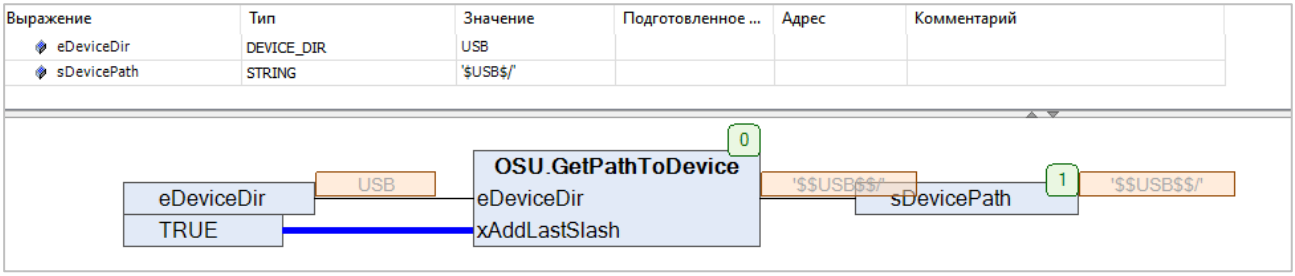


Рисунок 3.11.2 – Пример использования функции GetPathToDevice на языке CFC

Таблица 3.11.4 – Описание элементов перечисления DEVICE_DIR

Название	Тип	Значение	Описание
ROOT	UINT	0	Рабочая директория (**/PlcLogic**)
USB	UINT	1	Корневая директория USB-накопителя
SD	UINT	2	Корневая директория SD-накопителя
FTP	UINT	3	Директория FTP-сервера
VISU	UINT	4	Директория сервера web-визуализации (**/visu**)

Элементы **USB**, **SD** и **FTP** поддерживаются только для контроллеров ОВЕН с версией прошивки **1.2.0623.1009** или выше.

3.11.3 Функция GetDateFromSerialNumber

Функция **GetDateFromSerialNumber** возвращает дату производства прибора ОВЕН, определенную на основании его серийного номера, переданного на вход **sInput**. Так как в заводском номере не содержится информация о дне выпуска, то в качестве дня всегда используется «01».

Таблица 3.11.5 – Описание входов и выходов функции GetDateFromSerialNumber

Имя переменной	Тип	Описание
Входные переменные		
sInput	STRING	Серийный номер прибора ОВЕН
Выходные переменные		
GetDateFromSerialNumber	DATE	Дата производства прибора

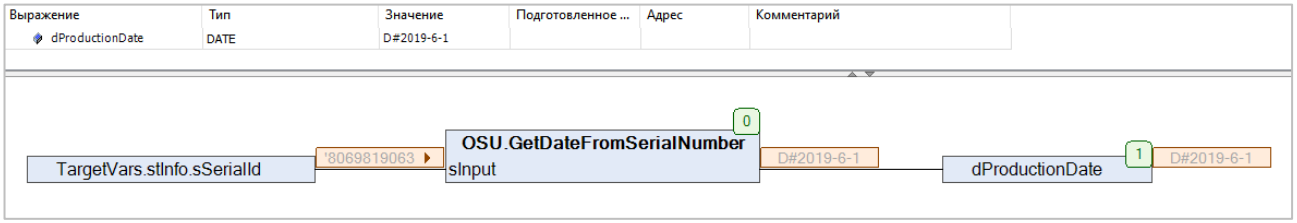


Рисунок 3.11.3 – Пример использования функции GetDateFromSerialNumber на языке CFC

Приложение A. Заполнители формата времени

Заполнитель	Описание	Пример отображения	Используется в функциях
d	День в виде числа (1–31)	8	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
dd	День с ведущим нулем (01–31)	08	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
M	Месяц в виде числа (1–12)	8	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT
MM	Месяц с ведущим нулем (01–12)	08	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT
y	Год века (0–99)	8	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT
yy	Год века с ведущим нулем (00–99)	08	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT
yyyy	Год	2008	DT_TO_STRING_FORMAT, DATE_TO_STRING_FORMAT
HH	Час в 24-часовом формате (01–24)	08	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
hh	Час в 12-часовом формате (01–12)	08 (и для 8-00, и для 20-00)	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT,
m	Минуты (0–59)	8	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
mm	Минуты с ведущим нулем (00–59)	08	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
s	Секунды (0–59)	8	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
ss	Секунды с ведущим нулем (00–59)	08	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
ms	Миллисекунды с ведущим нулем (000–999)	008	TOD_TO_STRING_FORMAT, TIME_TO_STRING_FORMAT
t	Идентификатор для 12-часового формата: A (часы < 12) и P (часы > 12)	A (для 8 часов)	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT
tt	Идентификатор для 12-часового формата: AM (часы < 12) и PM (часы > 12)	PM (для 15 часов)	DT_TO_STRING_FORMAT, TOD_TO_STRING_FORMAT